12-14-2012 12:00 AM

Automatic Classification of Epilepsy Lesions

Junwei Sun
The University of Western Ontario

Supervisor
Dr. Olga Veksler
The University of Western Ontario

Graduate Program in Computer Science
A thesis submitted in partial fulfillment of the requirements for the degree in Master of Science

AUTOMATIC CLASSIFICATION OF EPILEPSY LESIONS

(Thesis format: Monograph)

by

Junwei Sun

Graduate Program in Computer Science

A thesis submitted in partial fulfillment
of the requirements for the degree of
Masters of Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

THE UNIVERSITY OF WESTERN ONTARIO

School of Graduate and Postdoctoral Studies

CERTIFICATE OF EXAMINATION

Examiners:

.....

Dr. Terry Peters

Supervisor:

.....

Dr. Olga Veksler

.....

Dr. John Barron

.....

Dr. Mahmoud El-Sakka

The thesis by

Junwei Sun

entitled:

Automatic Classification of Epilepsy Lesions

is accepted in partial fulfillment of the
requirements for the degree of
Masters of Science

.....

Date

.....

Chair of the Thesis Examination Board

Acknowledgements

I would like to express my sincere thankfulness to my supervisor Dr. Olga Veksler for her constant support of my M.Sc. study, for her patience, enthusiasm and kindness. She gave me the confidence to explore my research interests and the assistance to overcome difficulties. This thesis would not have been possible without her guidance and help.

I would like to thank Dr. Lena Gorelick for her help with this research. She gave me many valuable suggestions on this work.

Also, I would like to thank Dr. Ali Khan for his cooperation and help on this research.

I would like to thank Dr. Terry Peters, Dr. John Barron and Dr. Mahmoud El-Sakka to be the examiners of this thesis. I would like to thank Dr. Mahmoud El-Sakka for his valuable comments on my thesis proposal.

I would like to thank Dr. Yuri Boykov and Dr. Steven Beauchemin. I had a great time attending their classes and learned much.

I would like to thank staff members from Computer Science Department for their assistance.

I am especially grateful for all my friends. They accompany me go through good and bad times. I greatly value their friendship.

Last but not least, I would like to thank my family for their unconditionally love. They always support me no matter how far away they are. I give my deepest gratitude to my parents' dedication and their endless love, support and encouragement.

Abstract

Epilepsy is a common and diverse set of chronic neurological disorders characterized by seizures. Epileptic seizures result from abnormal, excessive or hypersynchronous neuronal activity in the brain. Seizure types are organized firstly according to whether the source of the seizure within the brain is localized or distributed. In this work, our objective is to validate the use of MRI (Magnetic Resonance Imaging) for localizing seizure focus for improved surgical planning. We apply computer vision and machine learning techniques to tackle the problem of epilepsy lesion classification. First datasets of digitized histology images from brain cortexes of different patients are obtained by medical imaging scientists and provided to us. Some of the images are pre-labeled as normal or lesion. We evaluate a variety of image feature types that are popular in computer vision community to find those features that are appropriate for the epilepsy lesion classification. Finally we test Boosting, Support Vector Machines (SVM) and the Nearest Neighbor machine learning methods to train and classify the images into normal and lesion ones. We obtain at least 90.0% of accuracy for most of the classification experiments and the best accuracy rate we get is 93.3%. We also automatically compute neuron densities. As far as we know, our work of performing histology image classification and automatic quantification of focal cortical dysplasia in the correlation study of MRI and epilepsy histopathology is the first of its kind. Our method could potentially provide useful information for surgical planning.

Keywords: Epilepsy Lesion Classification, Machine Learning, Computer Vision, AdaBoost, Support Vector Machines

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction to Our Problem

The goal of this thesis is to learn to detect histology images of brain that are associated with epilepsy lesions. Our objective is to validate the use of MRI for localizing seizure focus for improved surgical planning. MRI, which is short for Magnetic Resonance Imaging, is a medical imaging technique used in radiology to visualize internal structures of the body in detail. MRI makes use of the property of nuclear magnetic resonance (NMR) to image nuclei of atoms inside the body. Figure 1.1 shows an example of an MRI image of brain. MRI provides good contrast between the different soft tissues of the body, which makes it especially useful in imaging the brain, muscles, the heart, and cancers compared with other medical imaging techniques such as computed tomography (CT) or X-rays. MRI is the most important neuroimaging test in epilepsy because it shows more details of the brain's structure than does a CT scan. In clinical practice, MRI is used to distinguish pathologic tissue from normal tissue.

Figure 1.2 shows a normal MRI brain image and an image with epilepsy lesion [61]. In our work, we validate use of MRI by classifying abnormal samples from normal samples for localizing seizure focus. This is the first time such analysis is done in the correlation study of MRI and epilepsy histopathology.

The images we used for our work are obtained with medical imaging equipment and techniques. Medical imaging is the widely used technique to create images of the human body for clinical purposes or medical science. MRI is one of the most commonly used medical imaging modalities. In our work, we do not work on the whole profile of brain histology image, but on sliced tissue from the whole profile. Since the images we used came from resected tissue, we can consider it as a pathology problem. It is the ultimate goal to diagnose epilepsy auto-

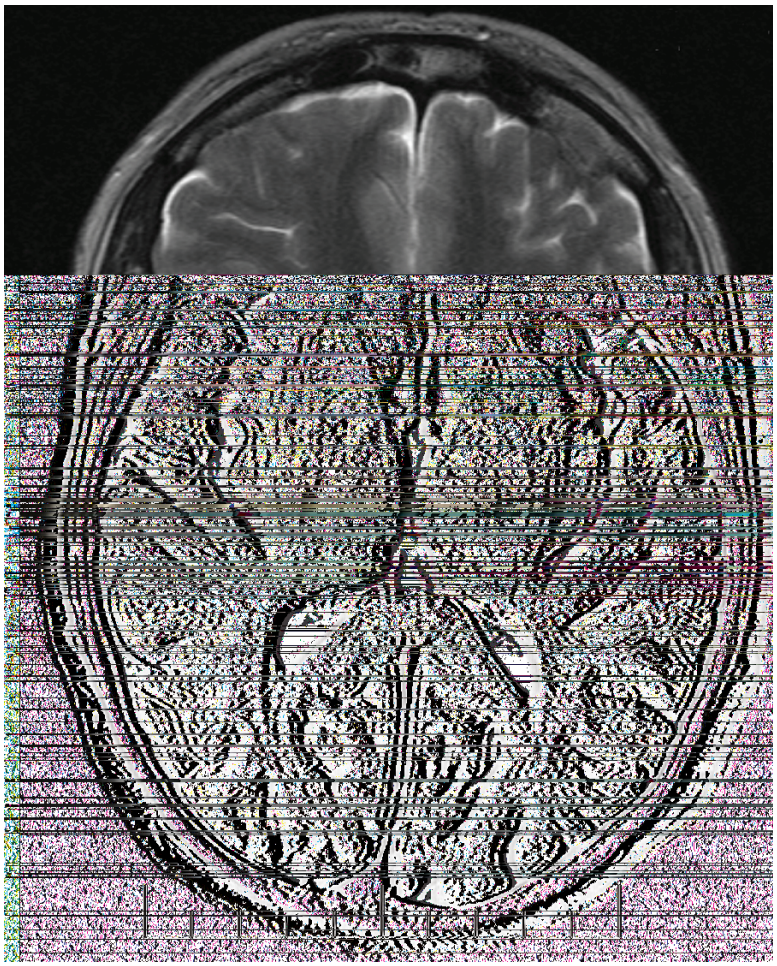


Figure 1.1: An MRI brain axial image.

matically from MRI and histology images of human brain, which requires correlation between MRI features and pathology. Hopefully it would be much easier for pathologists and doctors to diagnose and treat epilepsy patients in the future.

Figure 1.3 shows the overview of the acquisition and processing pipeline including pre-operative and post-operative MRI imaging and registration of MRI [21]. First, the recruited patients went through a series of MRI scans including structural, diffusion-weighted imaging, and relaxation mapping. In the following surgery, the resected tissue of the hippocampus and neocortex were provided to the pathology technologist for specimen imaging and histological processing. The specimen were immersed in a silicone-based lubricant and imaged using a gradient-insert with a 4 channel coil in a MRI scanner. MRI was performed on the specimens before and after overnight fixation in formalin. The next procedures on the specimens are accessioning, grossing, cut-plane identification, embedding in agar, then slicing into pieces.

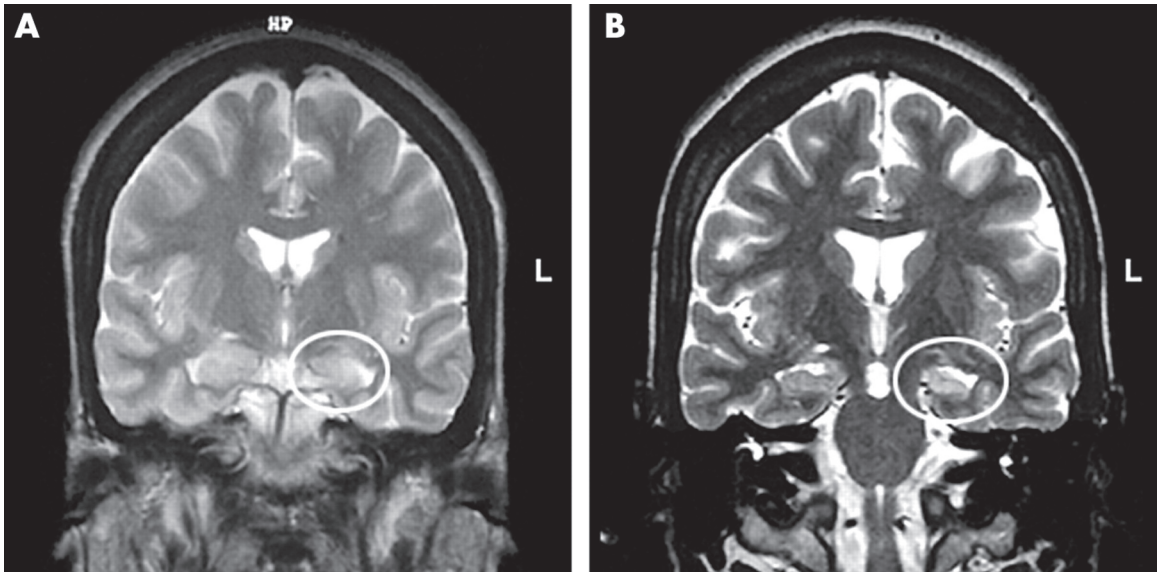


Figure 1.2: A is a normal MRI brain image and B is an MRI image with epilepsy lesion.

Then the specimens are sectioned, stained and digitized resulting in histology images. The histology images were also median filtered and centered in a frame [21]. The images we used for classification are histology images. Figure 1.4 shows a digitized histology image.

Figure 1.3: Overview of the acquisition and processing pipeline including pre-operative and post-operative MRI imaging and registration of MRI.

Seizure types are organized firstly according to whether the source of the seizure within the brain is localized or distributed. A partial seizure may spread within the brain by a process known as secondary generalization. Generalized seizures are divided according to the effect on the body but all involve loss of consciousness. Children may exhibit behaviors that are easily mistaken for epileptic seizures but are not caused by epilepsy. In our work, we are interested in classifying localizing seizures. The medical imaging scientists obtained histology images. We tried to find Focal Cortical Dysplasia (FCD) from those images taken from different epilepsy patients. FCD is the most common pathology in MRI-negative epilepsy and classification of FCD is needed for improved MRI validation. According to International League against

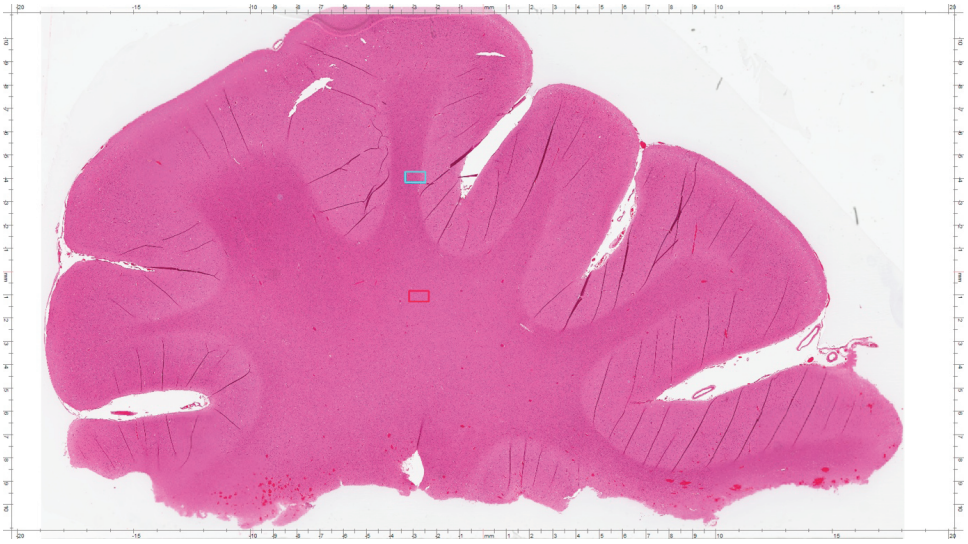


Figure 1.4: A digitized result in histology image.

Epilepsy (ILAE) [5], there are mainly three types of dysplasia. Type 1 FCD is the neuronal laminar malformations, which we attempt to find abnormalities from neuronal layering. FCD Type 2 is caused by dysmorphic neurons and it is more frequently encountered in extratemporal areas, particularly in the frontal lobe. The first two types are considered as isolated lesions. In contrast, the third type of FCD is associated with tumors or sclerosis. What we focused on in our work is a subtype of Type 1 FCD.

Type 1 FCD is mainly divided into two subtypes. Type 1a refers to radial abnormalities and Type 1b is tangential abnormalities. There is also a Type 1c which refers to FCD with both radial and tangential abnormalities. After obtaining digitized histology images, these abnormalities can be found manually by pathologists, but our goal is to detect these abnormalities and classify normal and lesion images automatically using standard machine learning techniques. Quantifying Type 1 FCD can have significant impact on histopathology correlation studies.

1.2 Introduction to Epilepsy Lesions

The cerebral cortex is a sheet of neural tissue that is outermost to the cerebrum of the brain. It covers the cerebrum and cerebellum, and is divided into left and right hemispheres. After the work of Korbinian Brodmann (1909) [16], the neurons of the cerebral cortex are grouped into six main layers. Each layer has a different composition in terms of neurons and connectivity.

The layered structure of the mature cerebral cortex is formed during development. The different cortical layers each contain a characteristic distribution of neuronal cell types and connections with other cortical and subcortical regions. The layers do not simply stack one over the other; there are characteristic connections between different layers and neuronal types. We want to classify normal and lesion samples by finding disorders in neuron layers. Figure 1.7 is a cortex drawing of a Spanish pathologist, Nobel laureate, Santiago Ramon y Cajal in 1905. Santiago Ramon y Cajal is considered by many to be the father of modern neuroscience. The cortex shows 6 neuron layers clearly.

Epilepsy is a set of chronic neurological disorders characterized by seizures. Factors like brain trauma, strokes, brain cancer, and drug and alcohol misuse may be associated with the causes of seizures. Brain lesions are important causes of epilepsy where there is scar tissue or another abnormal mass of tissue in an area of the brain. FCD is the most common pathology in MRI-negative epilepsy. In this section we will introduce the FCD that we are interested in and also the image registration of MRI images. Focal cortical dysplasias are frequently associated with epilepsy in both children and adults. They were first described in detail by Taylor et al. (1971) [52]. The term FCD was widely used since then. FCDs could be present in any location of the cortex. They have variable size and location, and may also affect multiple lobes. Different kinds of classification of FCD have been proposed. The classification system which is widely used now is from the work of Palmini et al. in 2004 [39]. Figure 1.5 and Figure 1.6 show the two subtypes of Type 1 FCD that we are interested in.

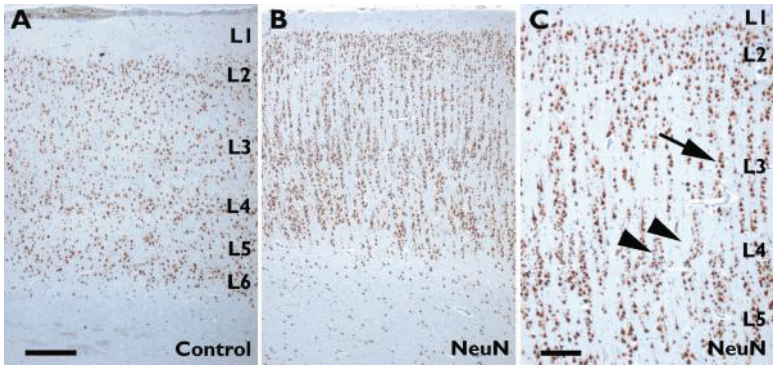


Figure 1.5: Examples of FCD Type 1a. Image A shows the normal appearing neocortex adjacent to the lesion shown in B and C. In image B, distinct microcolumnar arrangements can be detected in FCD Type 1a. At the arrows in sample C, there are abundant “microcolumnar” organization. A microcolumn is defined as more than 8 neurons aligned vertically.

Type 1a FCD is related to radial abnormalities. We can see from Figure 1.5 that at the

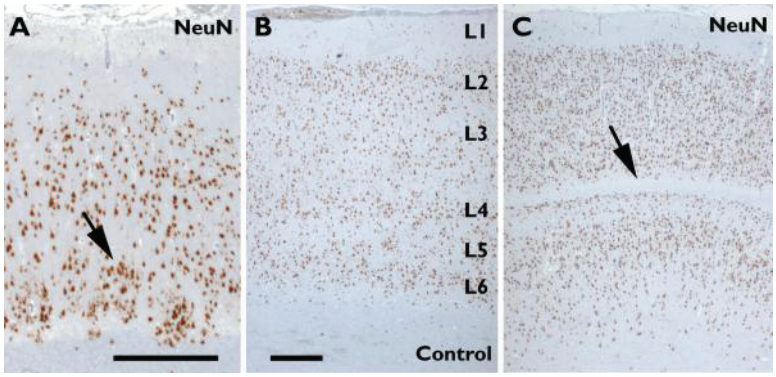


Figure 1.6: Examples of FCD Type 1b. There is no recognizable layering in sample A. In sample B, there is depletion of layer 2 and layer 4. In the sample C, there is a layer missing, which is supposed to be layer 4.

arrows, there are abundant "microcolumnar" organization. A microcolumn is defined as more than 8 neurons aligned vertically. The microcolumns can also be seen in non-epileptic brain samples, but with lower frequency. Type 1b refers to tangential abnormalities. The histopathologic definition of Type 1b FCD is loss of layering. In Figure 1.6, there is no recognizable layering in sample A. In sample B, there is depletion of layer 2 and layer 4. In sample C, it is obvious that there is a layer missing, which is supposed to be layer 4. Unless the area of FCD is large, patients do not have severe neurologic deficits and the main clinical manifestation is epilepsy. Seizure semiology depends on the location of lesion and patients with both Type 1 and Type 2 dysplasias present high seizure frequency.

Goubran et al. [21] described the processing to obtain the digitized the histology images we used in the experiments. The processing includes pre-operative and post-operative MRI imaging, histological processing and registration of histology and MRI. The pipeline is shown in Figure 1.3. They presented a novel pipeline for post-operative imaging of temporal lobe specimens, and evaluate methods for bringing these images into spatial correspondence. Their work allows for the spatially-local comparison of histology with post-operative MRI and paves the way for eventual registration with pre-operative MRI images.

The correlation of MRI with histopathology is necessary to understand the basis of MRI abnormalities and subsequently predict histopathology from in vivo MRI. Similar work has been done of comparing histology and MRI. Eriksson et al. [11]. tried to prove if particular quantitative MR parameters were associated with particular histological features. They suggested that with superior signal to noise ratios, novel quantitative MRI measures may generate data that correlate with histopathological measures.

1.3 Our Approach

The work in this thesis is based on the tools developed in computer vision and machine learning communities. The goal of computer vision is to understand the visual world, i.e. being able to analyze images, video sequences, etc. It is really difficult for a computer to duplicate the abilities of human vision. However, in some scenarios it is important and, indeed, the only option to let the camera and computer to automatically do the work of event detection [27], video tracking [4], object recognition [32], scene reconstruction [28], object modeling [42], etc.

Increasingly, especially in the last decade, computer vision research relies on the techniques developed in the machine learning community [18] [38] [41]. Machine learning is an area of computer science that seeks to develop algorithms that learn to accomplish some task through labeled (and sometimes unlabelled) data. Computer vision is full of problems like that. There have been many successes in computer vision using machine learning techniques, especially in the area of object detection [59] [31] and classification [25] [24] [17]. In our work, we approach the problem of epilepsy lesion classification in the machine learning framework.

Datasets of digitized histology brain images from 3 different patients were obtained by medical imaging scientists and provided to us. Some of the samples show abnormalities that are considered as Type 1 FCD by pathologists. There are three datasets: CorticalProfiles_NeuN, corticalQuantHist and H&E&NEUN stain. The datasets correspond to different locations of the cortex and have normal or abnormal labels. Figure 1.8 shows one normal sample and one abnormal sample from the corticalQuantHist dataset. The two samples are relatively distinct. The normal one has clear neuron layering and the abnormal one has a layer missing.

We extracted features from the samples, both global images features and local image features. Inspired by Lazebnik et al.'s work of spatial pyramid image representation[29], we divided the sample images into uniform layers, extracted features from each layer and then combined them to construct a feature vector. Then we used common machine learning methods to train classifiers. We used common machine learning methods for training. For the datasets that are not stained, our method can give at least 90.0% of accuracy rate for classification. The best result we obtained on the corticalQuantHist dataset is 93.2%. We also automatically detect neurons and compute neuron densities. The details of our work will be introduced in the following chapters.

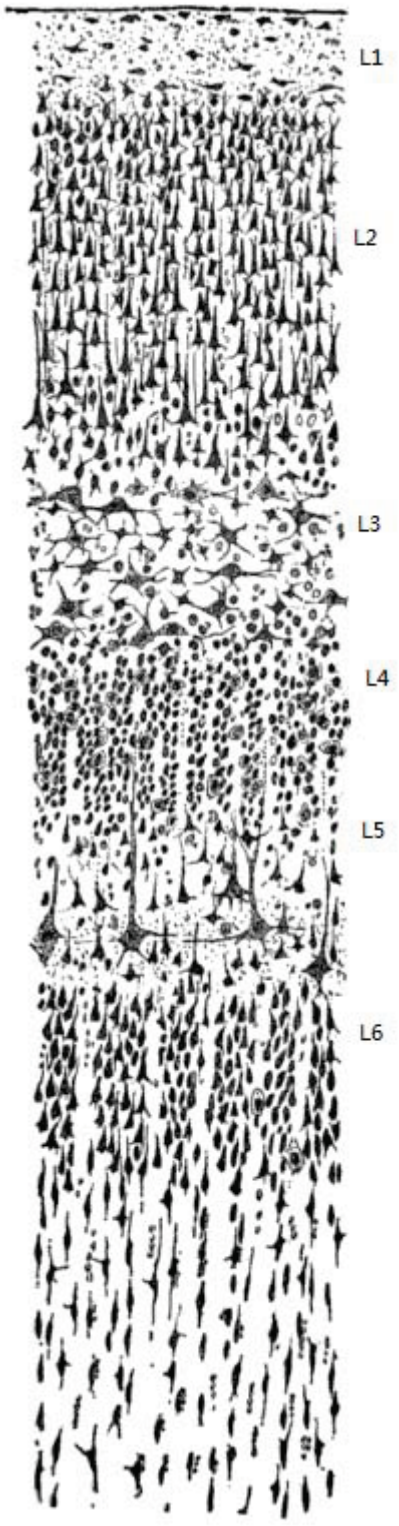
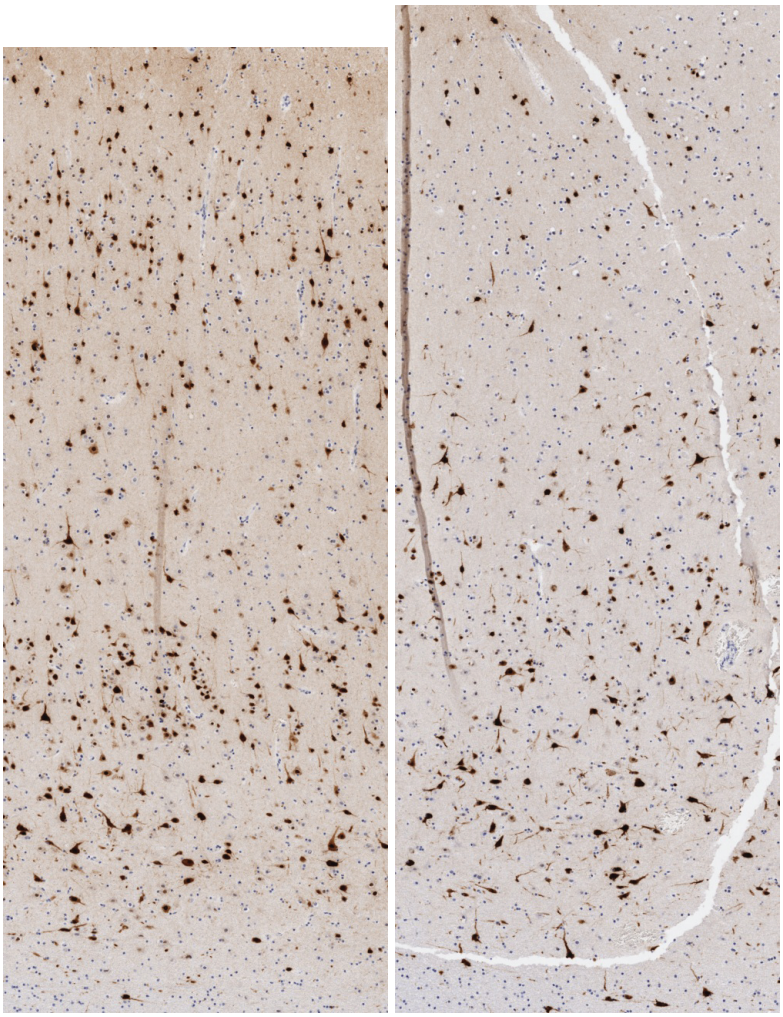


Figure 1.7: A cortex drawing of a Spanish pathologist, Nobel laureate, Santiago Ramon y Cajal in 1905. The cortex shows 6 neuron layers clearly.



(a) (b)

Figure 1.8: Two samples from the dataset. They are distinguishable in neuron layering. (a) is a normal sample. We can clearly see the neuron layering. (b) is an abnormal sample. It is obvious that the layer 2 is missing.

Chapter 2

Machine Learning

In this chapter, we will discuss topics about machine learning. In our work we solve a computer vision problem in the machine learning framework. We will talk about some common methods of machine learning and some applications of machine learning in computer vision problems. First we will give an introduction to machine learning. We will talk about different types and applications of machine learning methods. Then we will talk about specific machine learning methods we used in our work, Adaptive Boosting and Support Vector Machines. Afterwards we will discuss cross-validation used for estimating a method's performance. Finally, we will introduce many applications of machine learning in computer vision and other fields.

2.1 Introduction to Machine Learning

Machine learning is about designing and developing algorithms that can automatically improve the performance using example data or past experience. In 1959, Arthur Samuel defined machine learning as a "Field of study that gives computers the ability to learn without being explicitly programmed" [43]. Machine learning is a scientific discipline concerned with the development of algorithms that take as input empirical data. Machine learning is possible because computers have large data storage, large memory to handle the data and great computational power for calculating.

Machine learning is used when humans' expertise does not exist or humans are unable to explain their expertise. For example, it is very difficult to write a program that can automatically recognizing a face, because even though we do know a successful system which accomplishes this task with a high degree of accuracy, namely the human brain, we don't know exactly how our brain does it. To write a program that completes the brain's procedure of rec-

ognizing a face is impossible. We need to give data to computers with correct output for an input. Machine learning algorithms take examples with labels and produce a program that does the job. Machine learning is also needed when solution changes in time or solution needs to be adapted to particular cases.

The world is driven by data. Data are recorded from some real-world phenomenon. In many problems, data are cheap and abundant but knowledge is expensive and scarce. Machine learning algorithms obtain patterns through analysis of the features of data, and employ the patterns to make predictions based on the new data. After looking for relations between given input data (also called training data), the algorithms employ the knowledge to make decisions on the new input data (testing data). Machine learning could be really difficult. The set of all possible behaviors of all inputs can be too large, so we have to generalize from the inputs to find an efficient way to get a useful output. Machine learning aims to learn general models from particular examples and build a model that is a good and useful approximation to the data.

There are three types of machine learning, supervised learning, unsupervised learning and reinforcement learning. Supervised learning is the machine learning task of generating a function from labeled training data. In supervised learning we are given the training examples of inputs and corresponding outputs and we want to produce the right outputs. The outputs of the training examples are often manually labeled by a human. Like the classification problem in our work, we are given the input images of training data and also the output normal or lesion labels marked by pathologists.

There are a few steps to perform in order to solve a supervised learning problem. The first step is to gather a training set. A set of input objects and corresponding outputs are gathered. The outputs can be either from human experts or from measurements. The next step is representation of input objects into feature vectors, that is, transforming the input object into a number of features that are descriptive of the object. For example, if the input is an image, then its representation might just be a vector with each component being the intensity of a particular image pixel, that is a feature vector is just the intensities of all image pixels, piled, for example, in a row-wise fashion. Let the training set be $(x_1, y_1), ..., (x_k, y_k)$, where x_i is the feature vector of sample i and y_i is its corresponding output. The next step is to choose the learning function and learning algorithm.

A function $f(x, t)$ is chosen to find the correct output y_i for the input x_i. The t in the function is a vector of control parameters. The training stage consists of adjusting parameters t until the output $f(x_i, t)$ is close to the desired output y_i as much as possible for as many samples as possible. This is usually done by some optimization algorithm that is able to search over

parameter space t efficiently. To evaluate the accuracy of the learned function, we run $f(x,t)$ with the learned parameters t on examples that were not part of the training set. These examples are called training examples and the error on training examples is called the training error.

Type of Learning / Output	Supervised Learning	Unsupervised Learning
Discrete	Classification	Clustering
Continuous	Regression	Dimension Reduction

Figure 2.1: The subtypes of supervised learning and unsupervised learning.

Figure 2.1 shows the subtypes of supervised learning and unsupervised learning. There are two types of supervised learning, classification and regression. In regression, output labels y_i are real numbers. In classification, the output labels y_i are restricted to lie in a finite set, usually a set of integers, where each integer index a "class" of interest. Thus output of a learning function $f(x,t)$ is an integer indicating the class that x belongs to, according to $f(x,t)$. In classification, $f(x,t)$ is often called a "classifier".

A binary classification problem is a problem with 2 classes. Some cases of binary classification problems are linearly separable. Suppose the dimension of the data is n, a dataset is linearly separable if $\exists t$, such that $t_0 + \sum_j t_j x^j > 0$, if $x = (x^1, ..., x^n)$ is a positive example; $t_0 + \sum_j t_j x^j < 0$, if $x = (x^1, ..., x^n)$ is a negative example. Figure 2.2 shows a linearly separable binary classification problem. Figure 2.3 shows another example of a binary classification problem on 2-dimensional data. This example is not linearly separable. Linear classification is a very well understood and researched problem, with many algorithms for finding optimizing for parameters t. However, many practical problems are not linearly separable.

To design a classifier, we need several steps shown in Figure 2.4. Let x be the input sample from the training set and y be its correct label (output). The goal is to find the best parameter t to fit the function $y = f(x,t)$ with minimized prediction error over all samples in the training set. In classification problem, usually the error is defined as the number of misclassified examples. First we need to gather a training set. For the learning to be successful, it is very important that the training set is as representative as possible of the examples that will be encountered at

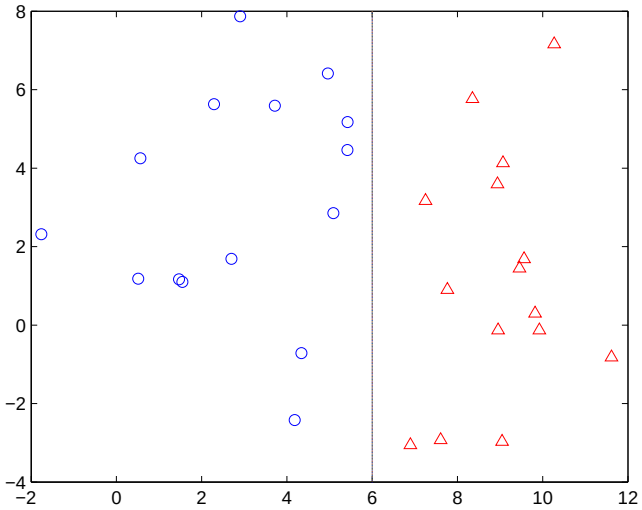


Figure 2.2: An example of a linearly separable binary classification problem on 2-dimensional data.

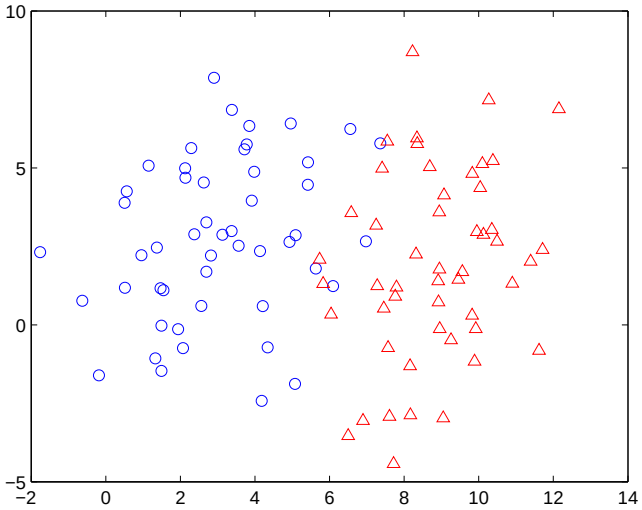


Figure 2.3: An example of a binary classification problem on 2-dimensional data.

the application, i.e. testing stage. If the training and testing datasets differ significantly, good performance on the training data may not transfer to the testing data.

The next step is feature extraction. We should analyze the data to find distinguishing features that could separate samples from different classes. A lot of work is usually done at this

stage. Good features can lead to a linearly, or almost linearly separable classification problem that admits good learning algorithms. Bad features may require more sophisticated learning machines $f(x,t)$ that are harder to train.

The third step is training the classifier. We should decide which model to use as a classifier. It is better to try simple classifier first. It is better to have "smart" features and simple classifiers than simple features and "smart" classifiers. In the training process, we change the parameter t so that it fits the data. We should try to avoid overfitting, which is explained below. Sometimes it is necessary to trade off the goodness of fit against the complexity of the model. After we have trained the classifier, the last step is to evaluate the classifier to see how well it performs on the data it has not seen so far.

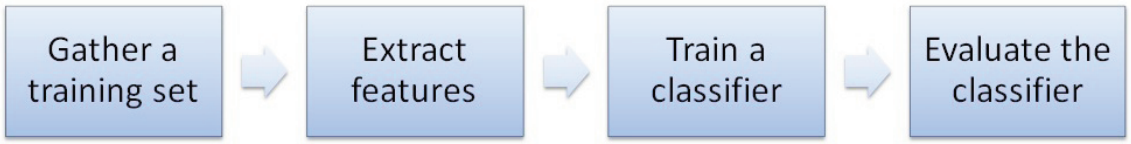


Figure 2.4: The steps for designing a classifier.

Choosing the values for the parameters that minimize the loss function to zero on the training data is not necessarily the best policy, especially if the classifier $f(x,t)$ is able to carve out arbitrary decision regions. It is highly possible that the classifier is overfitted. Overfitting means that model is too "complex" and fits irrelevant samples (noise) in the dataset. Figure 2.5 shows an example of overfitting. Although there are some misclassified samples in the classifier of Figure 2.5(a), it is a good classifier because the decision boundary is smooth and can separate most of the data correctly. Such classifiers with smooth decision boundaries are likely to generalize well. Generalization is the ability of a classifier to perform accurately on new, unseen examples after having been trained on training set. The classifier in Figure 2.5(c) is expected not to generalize well, since it is too tuned to particular data but not to the true model. For new data samples it is more likely to give high error rate. To avoid overfitting, it is necessary to use additional techniques like cross-validation that can indicate when further training is not resulting in better generalization.

Common classifiers include nearest neighbor, decision tree, SVM, boosting, etc. We will

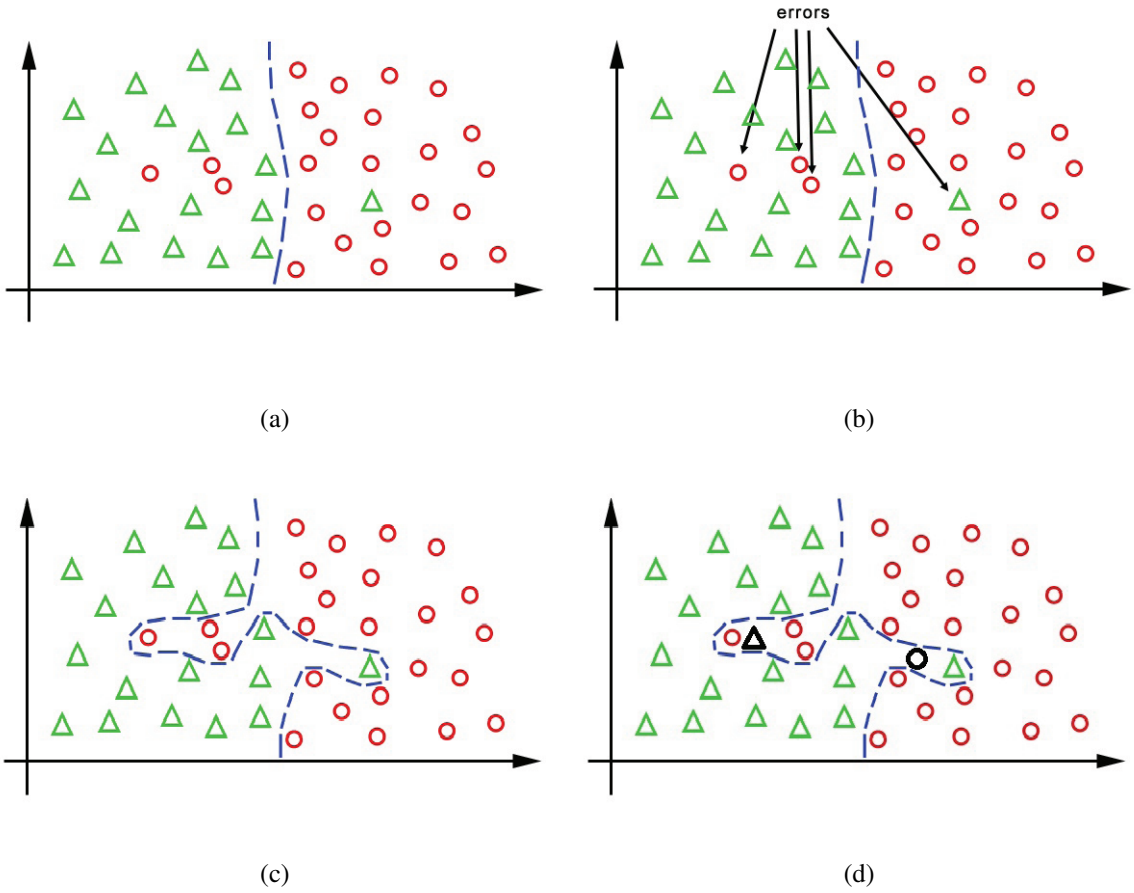


Figure 2.5: An example of overfitting. (a) is one classifier of a dataset. Some sample was misclassified as shown in (b). (c) is another classifier of the dataset. It seems to be a perfect classifier since no sample was misclassified. However, when the two black new data come, they are misclassified, as shown in (d).

introduce the classifiers in next sections. Classification has a wide spectrum of applications, including face recognition [54] [64], handwritten [30] or spoken words recognition [53], medical image diagnosis [55], unusual sequences of credit card transactions [2], recognizing spam emails [19], etc.

Regression is another type of supervised learning. It is a technique for estimating the relationships among variables. The output labels are continuous. The variables in a regression model contain the independent variables x, the dependent variable y and the unknown parameters t. A regression model relates y to a function of x and t. The goal is to best fit the function $y \approx f(x,t)$.

Linear regression is the most basic type of regression. In linear regression we try to fit a function $y = tx+b$. For a n-dimensional feature vector $x = (x^1, x^2, ..., x^n)$, the output function is

$f(x) = tx + b = \sum_{j=1}^{n} t_j x^j + b$. A simple linear regression problem is the least squares estimator of a linear regression model with one single covariant. Figure 2.6 shows an example of a simple linear regression of 1-dimensional.

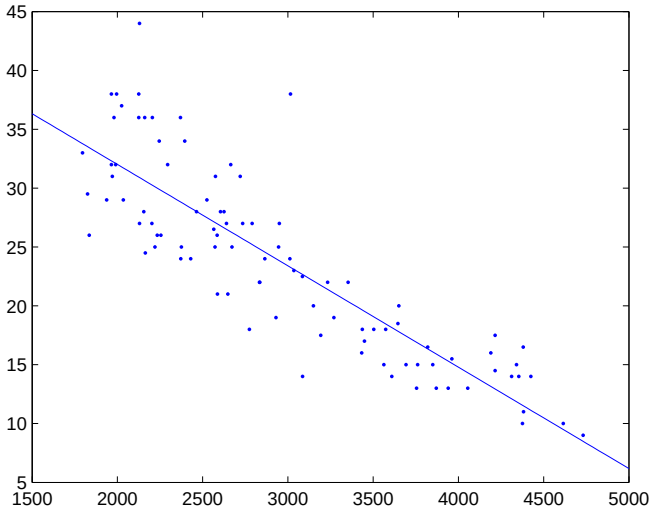


Figure 2.6: This is an example of a simple linear regression. The independent variable has 1 dimension.

Regression also faces the problem of overfitting. Figure 2.7 shows an example of an overfitted regression model.

Unsupervised learning is when we only have the training inputs and the goal is to find hidden structures in unlabeled data. There is no error or reward signal to evaluate a potential solution. Unsupervised learning is necessary because in many machine learning problems, big datasets do not come with labels since labeling the data could be expensive. Unsupervised learning includes clustering and dimension reduction. When the outputs are discrete, it is a clustering problem.

Clustering is a common example of unsupervised learning. In clustering, data are assigned into different clusters and samples in the same cluster should be more similar to each other than to samples in other clusters. To evaluate a clustering result, there are internal and external evaluations. An internal evaluation means the clustering result is evaluated based on the data that was clustered itself. An external evaluation are based on data that are not used for clustering. Figure 2.8 shows an example of clustering data into 3 clusters. Clustering can be used in problems dealing with unlabeled data, such as data mining [37], image segmentation [46],

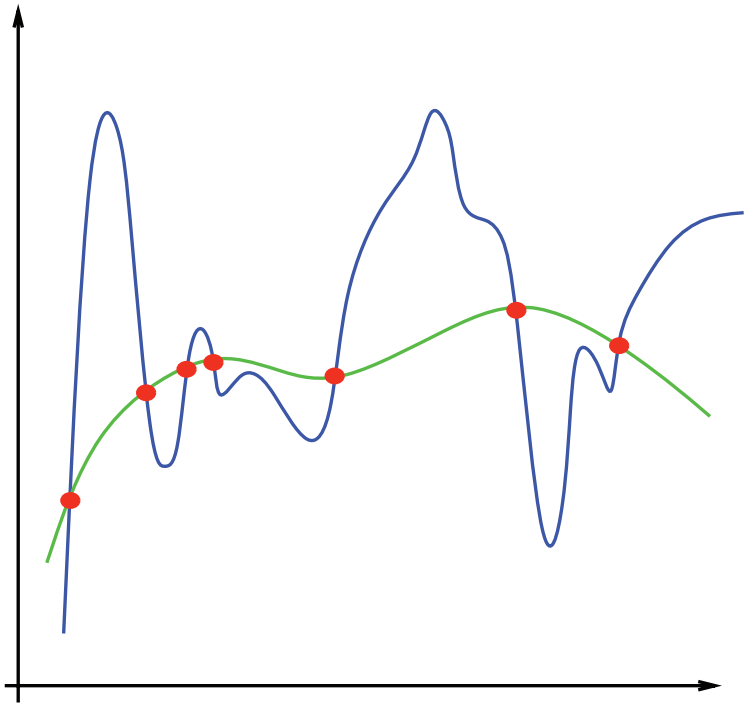


Figure 2.7: This is an example of an overfitted regression model. The green line is a better regression model. The blue model is overfitted.

recommender systems [1], medical imaging [3], etc.

K-means is a simple and fast clustering method [34]. K-means tries to partition the n samples $(x_1, x_2, ..., x_n)$ into k sets $S_1, S_2, ..., S_k$ to minimize the within-cluster sum of squares $\sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$. The μ_i is the cluster center of cluster S_i. The algorithm converges to a local minimum. We will describe k-means algorithm in Chapter 4.

The third type of machine learning is reinforcement learning which concerned with how an agent takes actions in an environment in order to maximize payoff. It is the objective to find out which actions are good based on the past experience. There is no supervised output in reinforcement learning but delayed reward. Its applications include credit assignment problem [50], game playing [26], robotics [51], etc.

Machine learning is a multidisciplinary field, combining cognitive psychology, neuroscience, mathematics and computer science. Machine learning has a wide spectrum of applications in computer science, including data mining, computer vision, natural language processing, bioinformatics, etc. The applications in computer vision include object detection [59], object classification [31], segmentation [40] and motion tracking [49], etc.

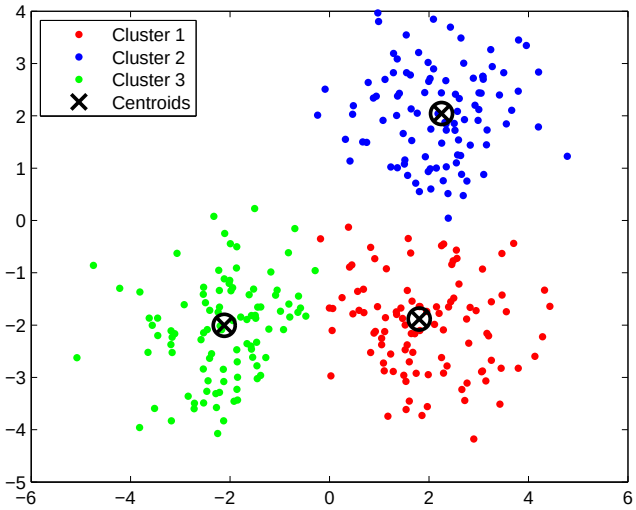


Figure 2.8: This is an example of clustering data into 3 clusters.

2.2 Nearest Neighbor

Nearest neighbor is a common and simple classifier in supervised learning. In the k-nearest neighbor method, a data point in the feature space is classified by the vote of its k closest neighbor points in training set. The distance metric includes Euclidean distance, Hamming distance, etc. A data point is classified to the most common class among its k neighbors. It can be useful to weight the contributions of the neighbors, so that the closer neighbors contribute more than the more distant ones. When $k = 1$, a data point is simply classified as the class of its nearest neighbor. Figure 2.9 shows an example of 3-nearest neighbor.

In the k-nearest neighbor, the choice of k is crucial. Different k will give different classification results. Theoretically, the larger the k, the better the classification. However, the theory assumes that there is an unlimited supply of training samples next to the point we want to classify. In practice, we never have unlimited supply of training data, so the best choice of k depends on the data. If k is too large, the k-nearest neighbor might oversmooth the decision boundaries. If k is too small, it will cause noisy decision boundaries. In a binary classification problem, it is better to choose k to be an odd number to avoid tied votes. A good k can be selected by heuristic techniques like cross-validation, which we explained in Section 2.5.

It can be proved that k-nearest neighbor classifier works very well when the amount of training examples is large. In fact, even with $k = 1$, the error rate can be shown to approach just twice worse than the optimal error rate [7]. However, k-nearest neighbor is computationally

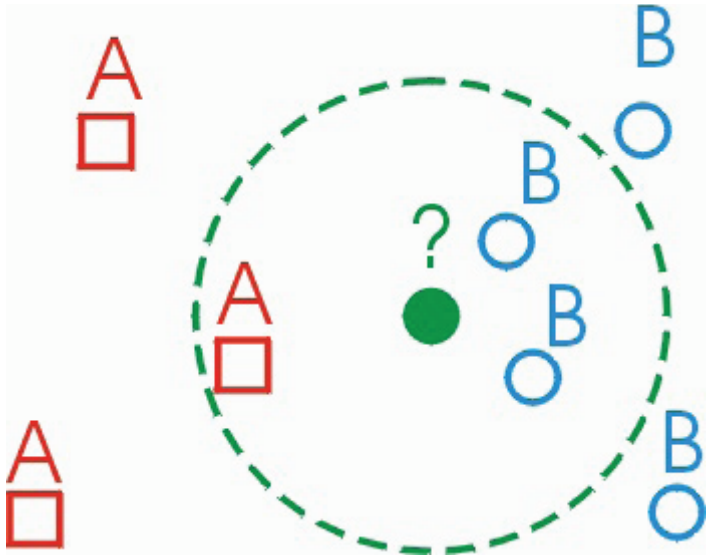


Figure 2.9: An example of 3-nearest neighbor. The green sample should be classified as class B.

expensive precisely when the number of training samples is large, or k is large. If the data has dimension of d and there are n data in training set, to compute one nearest neighbor for a data point requires $O(nd)$ time. To find k nearest neighbors the complexity is $O(knd)$.

2.3 Boosting

Boosting is a common machine learning algorithm which belongs to supervised learning. The main idea of boosting is to create a strong learner from some weak learners. It is hard to design a single accurate classifier that generates well. A weak learner is slightly related to true classification whose classification accuracy could be just better than random. A strong learner should be well-related to the true classification. When using boosting algorithms, weak learners are weighted in some way and combined into a final stronger classifier. In each step of boosting, a weak learner was added, and the samples that are classified correctly will have lower weight and those who are misclassified will get higher weight. Different weak classifiers will pay focus on different samples. There are many kinds of boosting algorithms. AdaBoost is the most popular boosting algorithm.

AdaBoost which stands for Adaptive Boosting, is a boosting algorithm formulated by Freund and Schapire [14]. The algorithm is adaptive because the subsequent classifiers are tweaked in favor of those examples misclassified by previous classifiers. The algorithm stops

when it reaches the maximum number of iterations or when the weighted error rate given by subsequent weak learner is close to random. In many problems, the number of data features is huge, it would be really expensive if we use all the features for classification, so it is more efficient to choose just the best features and still get the classification done. Generally, a weak learner corresponds to a small set of features, AdaBoost will collect those weak learners and combine them into a strong learner. In this way, AdaBoost automatically select the features that are more useful.

AdaBoost is an iterative function. After T iterations, it produces the discriminant function $g(x) = \sum_{t=1}^{T} \alpha_t f_t(x)$, the $f_t(x)$ is the t-th weak classifier and α_t is its relative weight. The AdaBoost algorithm for binary classification is shown in Algorithm 1.

Algorithm 1 The AdaBoost for a binary classification

Given:

• a training set with N samples: $(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

• number of iterations T

Initialize weights of each sample $d_1(i) = \frac{1}{N}, i = 1, ..., N$.

for $t = 1, ..., T$: **do**

• Find the classifier $f_t(x)$ that best classified positive and negative samples. The error rate is computed as $\varepsilon_t = \sum_{i}^{N} d_t(x_i) I(y_i \neq f_t(x_i))$. I is the indicator function:

$$I(y_i \neq f_t(x_i)) = 1, \text{ if } y_i \neq f_t(x_i); I(y_i \neq f_t(x_i)) = 0, \text{ if } y_i = f_t(x_i).$$

Choose the $f_t(x)$ with the lowest ε_t among the weak classifiers.

• Assign $\alpha_t = \log \frac{1-\varepsilon_t}{\varepsilon_t}$

• Update sample weights $d_{t+1}(i) = d_t(i) \times \exp[-\alpha_t y_i f_t(x_i)]$

Normalize weights

$$d_{t,i} \leftarrow \frac{d_{t,i}}{\sum_{j=i}^{N} d_{t,j}}$$

end for

The final hypothesis function:

$$f_{FINAL}(x) = sign[\sum_{t=1}^{T} \alpha_t f_t(x)]$$

Figure 2.10 shows an example of how the algorithm performs on a binary 2-dimensional classification problem [14].

AdaBoost is a fast and simple algorithm for performing supervised learning. It is sensitive to noisy data and outliers. The performance of AdaBoost depends on the data and weak learners. AdaBoost can fail if the weak learners are too "complex" and fit irrelevant samples in the

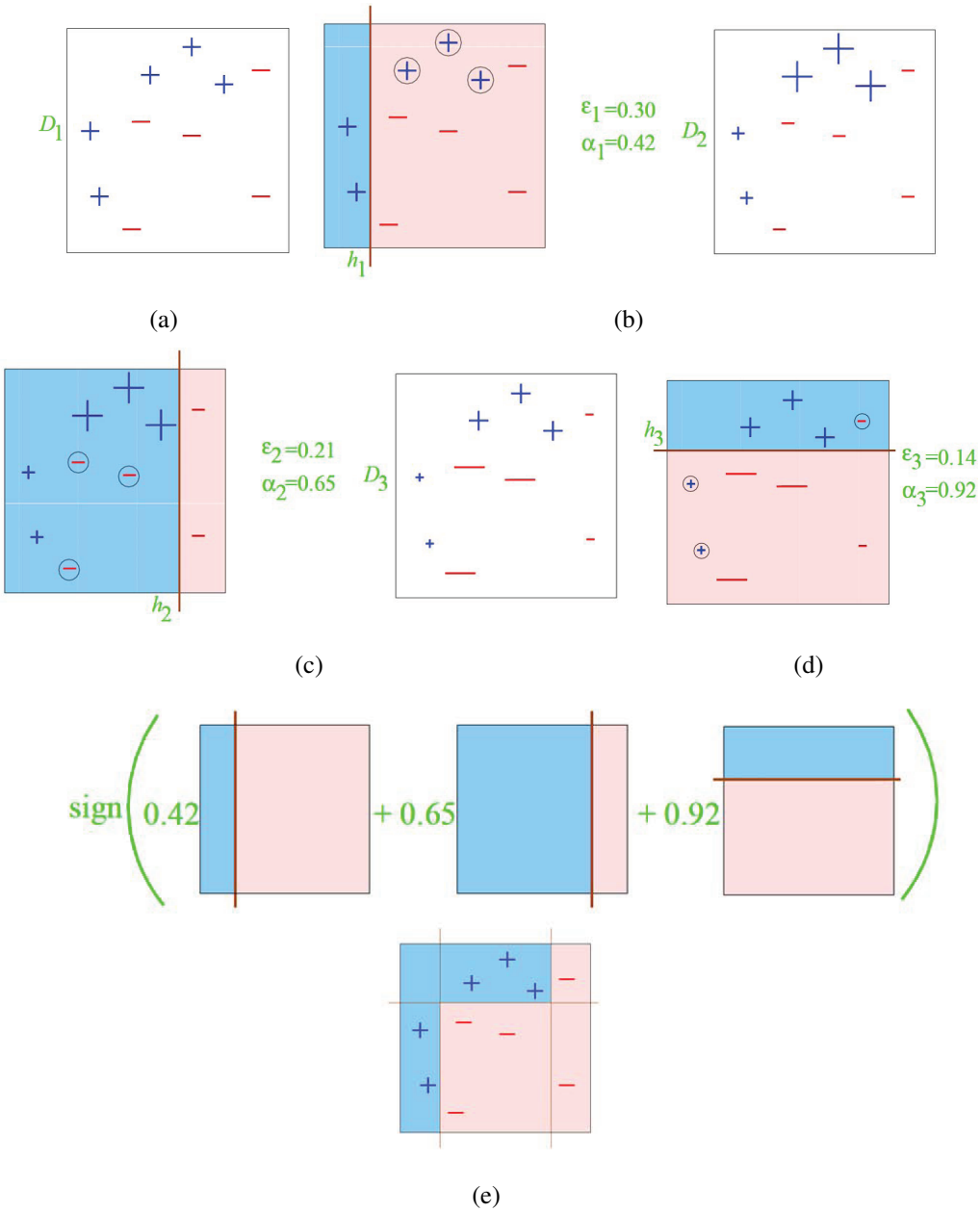


Figure 2.10: An example of the AdaBoost algorithm on a binary 2-dimensional classification problem. In (a), the samples are not linearly separable. (b), (c) and (d) show three iterations of AdaBoost, each time a new weak classifier is included. The misclassified samples get higher weights, marked by bigger positive or negative signs. In (e), the weak classifiers are combined together to form a strong classifier. The training error become 0.

dataset, which is overfitting. It can also fail if the weak learner are too weak, which will cause underfitting.

In our work, we used the GML AdaBoost toolbox implemented by Vezhnevets [57]. The

toolbox contains three different boosting schemes: Real AdaBoost [44], Gentle AdaBoost [15] and Modest AdaBoost [58]. In this toolbox, each weak learner corresponds to a decision tree which consists of a few nodes. Each node corresponds to a threshold of one feature dimension. Every weak learner gives the classification based on the thresholds on tree nodes. The three boosting schemes have different advantages. Real AdaBoost is the basic boosting algorithm introduced by Freund and Schapire [14]. Gentle AdaBoost is more robust and stable than Real AdaBoost, which performs better on noisy data and is more resistant to outliers. Modest AdaBoost is more resistant to overfitting.

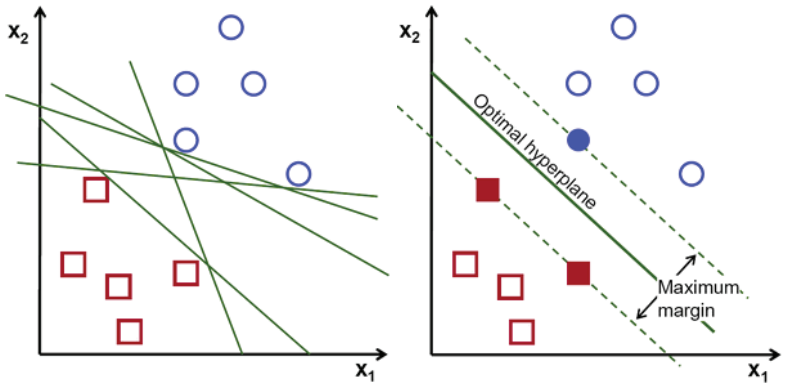
The use of boosting was popularized in computer vision by Viola et al.'s work [59]. In particular, they applied it successfully to the face detection problem, and also to pedestrian detection [60].

2.4 Support Vector Machines

Support vector machines are supervised models that analyze data and recognize patterns, used for classification and regression analysis. The algorithm was invented by Vapnik and Cortes [6]. SVM is used to solve the binary classification problem. SVM takes a set of input data and outputs of two possible classes, trying to train a classifier than can give correct prediction to new inputs.

In the binary classification problem, SVM constructs a hyperplane that can best separate data from the two classes by having the largest distance to the nearest training data of any class. Figure 2.11(b) shows the optimal separating hyperplane that maximize the distance to the nearest training data of both classes. The margin is twice of the distance from the nearest training data to the separating hyperplane.

The separating hyperplane can be described as a function $w^T x + b = 0$, where w is a weight vector, x is an input vector and b is the bias. If the training data are linearly separable, we can select two hyperplanes in a way that they separate the data and there are no points between them, and then try to maximize their distance. If we multiply w and b by the same positive factor, the hyperplane remains unchanged. We let $min|w^T \times x + b| = 1$. We define the support vectors for two points $w^T \times x_1 + b = +1$ and $w^T \times x_2 + b = -1$. The distance of a point x_i from a hyperplane is $\frac{w^T x_i + b}{||w||}$. $y_i = sign(w^T x_i + b)$ gives the output class of a data point x_i. The distance from support vector and the separating hyperplane is $\frac{1}{||w||}$. The margin is twice the distance, $\frac{2}{||w||}$. To maximize $\frac{2}{||w||}$ we can minimize $||w||$, or equivalently minimize $\frac{||w||^2}{2}$. So we have an optimization problem described as:



(a) (b)

Figure 2.11: An example of some bad separating hyperplanes shown in (a) and the optimal separating hyperplane on the same training set shown in (b).

$$\text{Minimize: } \frac{\|w\|^2}{2}$$
$$\text{Condition: } y_i(w^T x_i + b) > 1 \text{ for } i = 1, 2..., n$$

After optimization we obtain the hyperplane and use the $sign(w^T x_i + b)$ function for classification.

The idea behind SVMs is that a classifier with a large margin is likely to generalize better to the new data. Given any training sample, it is quite likely that there is another sample of the same class next to it that we don't observe in the training data but which will be possibly encountered in the future as a test sample. Intuitively, if a decision surface lies close to one of the training samples, then a nearby sample of the same class (that we don't observe in training) will lie on the wrong side of that separating surface. By moving separating surface as far as possible from all training examples, we are likely to minimize error on the examples that will occur in the future. This is just an intuitive explanation, but it can be shown theoretically that SVMs have good generalization properties.

If the training data points are not linearly separable, we can use the soft margin method. The soft margin method chooses a hyperplane that splits the examples as correctly as possible, while allowing some examples to be on the wrong side of the decision hyperplane. The idea is to still try to find the hyperplane with the largest margin, but now a small fraction of examples is allowed to lie inside the margin area. The number of such examples that are not in the ideal position is controlled by parameter C in the equation below. This method introduces slack variables ξ_i, which measure the degree of misclassification of example x_i. The optimization function turns into:

$$\text{Minimize: } \frac{\|w\|^2}{2} + C\sum_{1}^{n}\xi_i$$
$$\text{Condition: } y_i(w^T x_i + b) \geq 1 - \xi_i \text{ for } i = 1, 2..., n$$

This optimization is a trade off between a large margin and a small error penalty.

A nonlinear SVM lifts the training examples into a higher dimensional space $\Phi : x \rightarrow \phi(x)$. The idea is that in a higher dimensional space, the data are more likely to be linearly separable. See an illustration shown in Figure 2.12. When solving the optimization problem we need to compute $x_i^T x_j$. However, computing in high dimensional space is computationally expensive. The idea to avoid expensive computation is to use a kernel function $K(x, z) = \phi(x)^T \phi(z)$, this is so-called "Kernel Trick". For example, a Gaussian kernel is $K(x_i, x_j) = \exp(-\frac{|x_i - x_j|^2}{2\sigma^2})$.

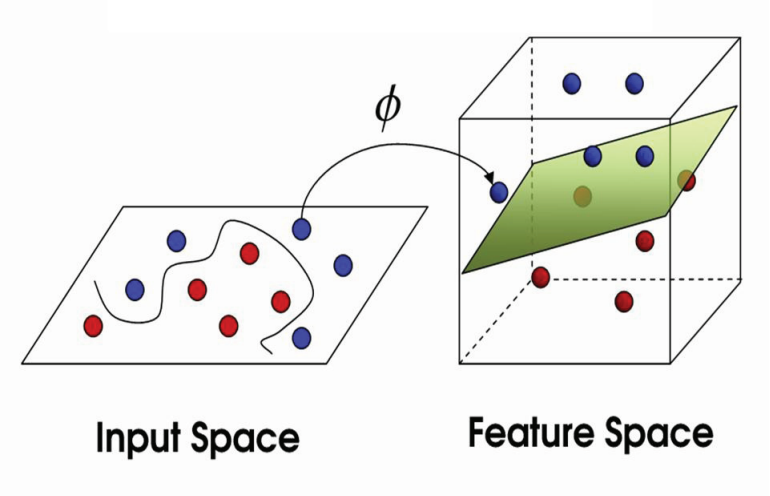


Figure 2.12: Samples are lifted to higher dimensional space, where they become linearly separable.

2.5 Cross-validation

In machine learning, cross-validation is a technique for estimating performance of a learning function $f(x, t)$ as well as choosing any parameters of the classifier that cannot be tuned on the training data. In our work, we use cross-validation to estimate performance of our classifiers.

When the collected labeled data are very large, we can afford to divide into the training and

testing sets. The training set is used to tune parameters t of the classifier and the testing set is used to test the performance of the classifier on the data it has not seen before.

However, if the dataset is not large, such a split into training and testing is wasteful. Indeed, to train a classifier, we need as large amount of data as possible, the more the better. Thus setting aside part of the data for testing means our classifier may not be very well trained. Cross-validation was designed to get a more accurate measure of performance and not to waste as much data as the traditional test/train data split.

The data are partitioned into k folds. We leave out the first fold and train on the remaining k-1 folds. After training, we compute the error on the first fold. In classification, the error is usually the number of misclassified examples, in regression, it's the sum of squared differences. Afterwards, we leave out the second fold and train on the remaining k-1 folds, and test on the fold we left out. We perform k iterations and average the errors from all iterations.

Cross-validation types are distinguished by how the subsets are partitioned. The common types of cross-validation include k-fold cross-validation, leave-one-out cross-validation and repeated random sub-sampling validation. In k-fold cross-validation, the samples are divided uniformly in to k subsets. The k-fold cross-validation can be repeated many times by randomly shuffling the data.

Leave-one-out cross-validation is similar to k-fold cross-validation; just the k is set to the number of samples. In leave-one cross-validation, each time a single sample is taken out as validation data and the other samples are used for training. Leave-one-out cross-validation can give more accurate estimation of a model, but it is computationally expensive.

Repeated random sub-sampling validation is another type of cross-validation. This method randomly divides the dataset into training and testing set, the error is assessed in the validation data of each split. The accuracy is averaged over the splits. The advantage of this method is the proportion of the training data and validation data is not dependent on the number of folds. However, the results will vary if the experiments are repeated on different random splits.

Chapter 3

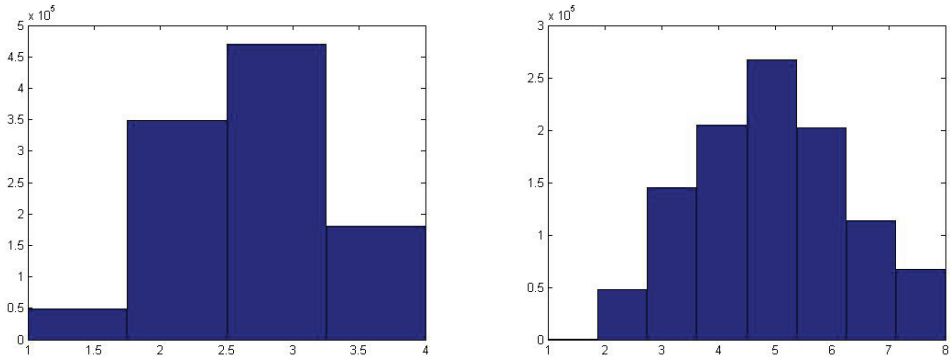
Image Features

In this chapter, we will describe the image features and feature representation method we used in our work. In computer vision, image feature refers to the specific structures in an image ranging from small as points or edges to big as objects. Image features can describe what is special or interesting in one image. There are global features which describe images as a whole and local features which represent image patches. For different computer vision problems, different image features are needed. Sometimes we may use more than one kind of feature. In this chapter we will discuss four kinds of image features that we evaluated for our classification problem.

3.1 Intensity

Intensity is the most basic feature for grayscale images. In a grayscale image, each pixel has a number which represents its shade of gray. For 8-bit grayscale images, the numbers vary from 0 to 255. The intensity features are often not used directly, but rather quantized into a smaller range, either for efficiency or higher discriminatory power. We divided the intensity value 0-255 into n bins. For example, when n is 4 we have 4 bins of 0-63, 64-127, 128-191 and 192-255. Then we count the number of pixels appear in each bin and we get an intensity histogram of n bins. Figure 3.1 and Figure 3.2 show an example of an original grayscale image and its intensity histogram with 4 bins and 8 bins. In the feature vector the histogram is represented by its actual number.

Figure 3.1: The original grayscale image.



(a) (b)

Figure 3.2: An example of original image and its histograms of intensity. (a) is its intensity histogram with 4 bins. (b) is its intensity histogram with 8 bins.

3.2 Stable Regions

Maximally Stable Extremal Regions (MSER) is a local image features mostly used in blob detection. This technique was proposed by Matas et al [35]. It is also used for stereo matching and object recognition. MSER has been proved to be a reliable region detector since it outperforms many other region detectors [36]. MSER algorithm extracts from an image a number of regions that are stable connected component of some level sets of the image. The definition of MSER is given below.

Definition *Image* I is a mapping $I : D \subset Z_2 \rightarrow S$. An external region is well defined on an image if:

1. S is totally ordered (reflexive, antisymmetric and transitive binary relations exist). 2. An adjacency relation $A \subset D \times D$ is defined.

Region Q is a contiguous subset of D. (That is for each $p, q \in Q$ there is a sequence $p, a_1, a_2, ..., a_n, q$, and $pAa_1, ..a_iAa_{i+1}, ..a_nAq.$, where xAy means that pixel x is adjacent to pixel y.)

Region Boundary $\partial Q = \{q \in D \backslash q : \exists p \in Q : qAp\}$, meaning that the boundary of Q is the set of pixels adjacent to at least one pixel of Q but not belonging to Q.

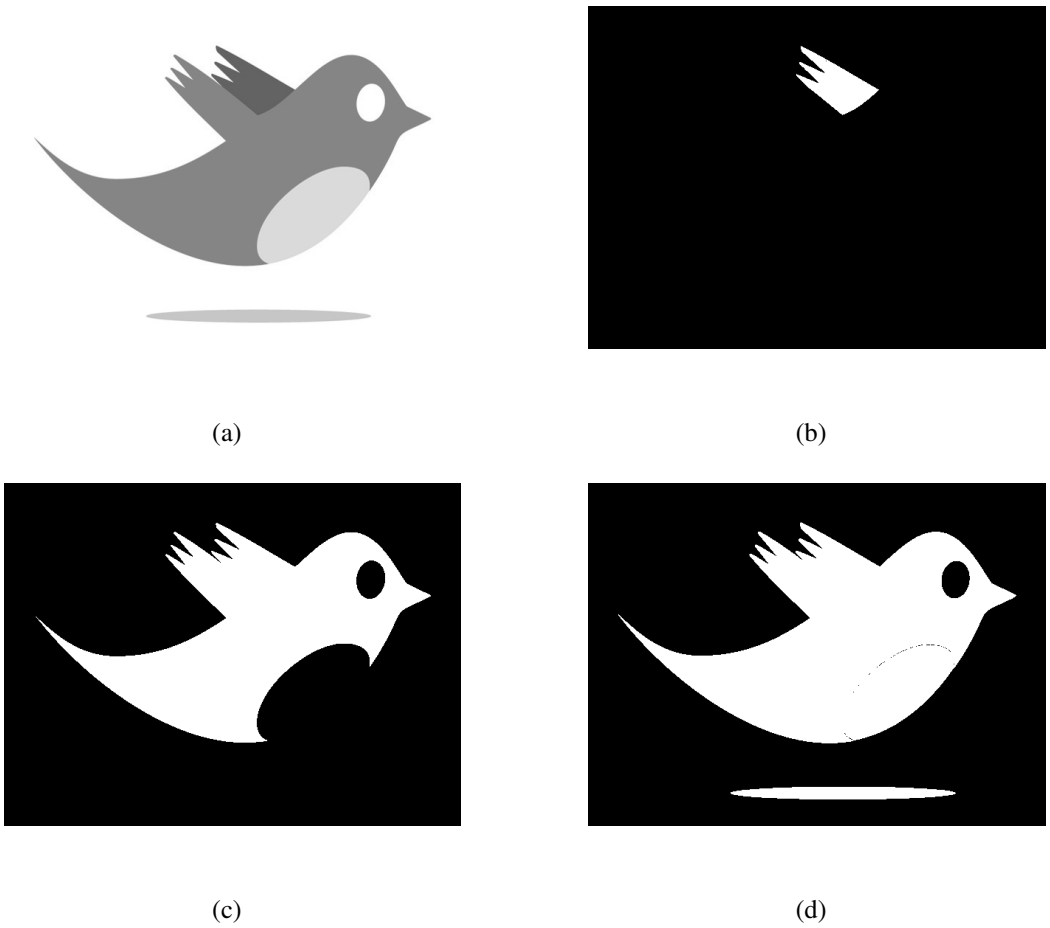
External Region $Q \subset D$ is a region such that either for all $p \in Q, q \in \partial Q : I(p) < I(q)$(minimum intensity region) or for all $p \in Q, q \in \partial Q : I(p) > I(q)$(maximum intensity region).

Maximally Stable Extremal Region Let $Q_1, ...Q_{i-1}, Q_i...$ be a sequence of nested extremal regions ($Q_i \subset Q_{i+1}$). Extremal region Q_{i*} is maximally stable if and only if $q(i) = |Q_{i+\delta} \backslash Q_{i-\delta}| / |Q_i|$ has a local minimum at i_*, where δ is a parameter of the method.

Intuitively, in an external region, the pixels in the region have higher or lower intensities than in the surrounding areas. Regions will change with different intensity range. We selected regions that are stable over intensity ranges. The concept of MSER can be explained by thresholding. When a threshold is given for an image, we can obtain a binary image with only white and black pixels. Namely all pixels with intensities below a threshold get mapped to 0, and all pixels above the threshold get mapped to 1. We can have a sequence of binary images for different thresholds t. The set of all connected components in the sequence of these binary images is the set of all extremal regions. Figure 3.3 shows an image and its thresholded binary images with different thresholds.

Based on thresholding, Donoser and Bischof [10] introduced an efficient way of detecting MSER. In Donoser and Bischof's work [10], MSER are detected using the component tree of threshold images, like those shown in Figure 3.5. Figure 3.4 is the input of their experiment. Each node of the component tree represents a connected region within the input image. The threshold levels changed to build the structure of the tree. A region is considered as stable if its size does not change significantly when threshold changes. In Figure 3.5, region 7 is detected as MSER because the size of it does not change much in the intensity range from 135 to 195.

Example of the MSER results with different parameters is shown in Figure 3.6 and Figure 3.7. We can see that there are some small regions detected inside the actual blob, since there are some bright pixels inside the blob and they are detected as maximum intensity re-



(a) (b)

(c) (d)

Figure 3.3: An example of original image and its thresholded binary images. (a) is the original image and the thresholds used in (b), (c) and (d) are 105, 165 and 225.

Figure 3.4: The input image.

gions. This issue can be solved by filling the big blob and removing the inside small regions. We will tell more about it in Chapter 4.

After we have detected the regions in one image, we can use the properties of the detected regions as image features, for example the number of regions, the size of regions, the size of bounding box of the regions, the intensity inside the regions, etc. The details of our MSER

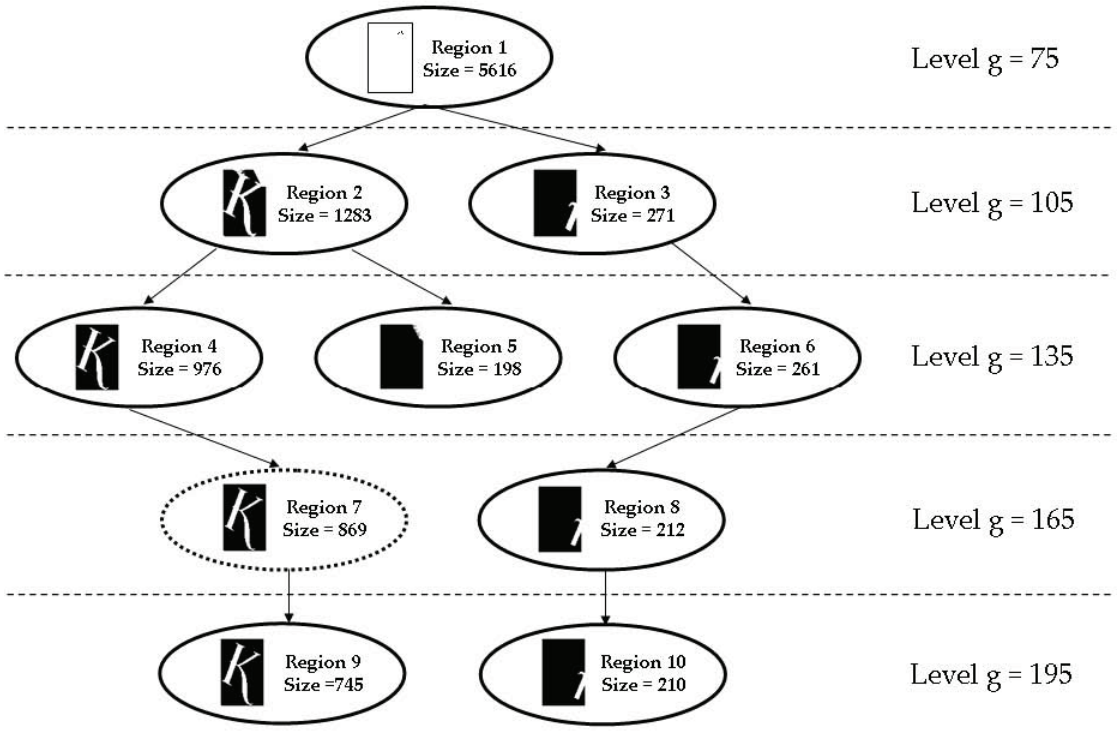


Figure 3.5: The component tree of threshold images of input Figure 3.4. Region 7 is detected as MSER because the size of it does not change much in the intensity range from 135 to 195.

feature extraction will be given in Chapter 4.

3.3 SIFT

Scale-invariant feature transform is an algorithm in computer vision to detect and describe local features in images. It was first published in 1999 [32] and has become one of the most significant feature detection techniques. SIFT feature is consistent with variations of the illumination, viewpoint and other viewing conditions [33].

In David Lowe's method[33], the first step is to detect stable keypoint locations by using scale-space extrema in the difference-of-Gaussian (DoG) function convolved with the image, $D(x, y, \sigma)$. The DoG function is computed from the difference of the nearby scales separated by a constant factor k. Figure 3.8 shows an approach to construct $D(x, y, \sigma)$. For each octave, the original image is repeatedly convolved with Gaussian filters to produce images separated

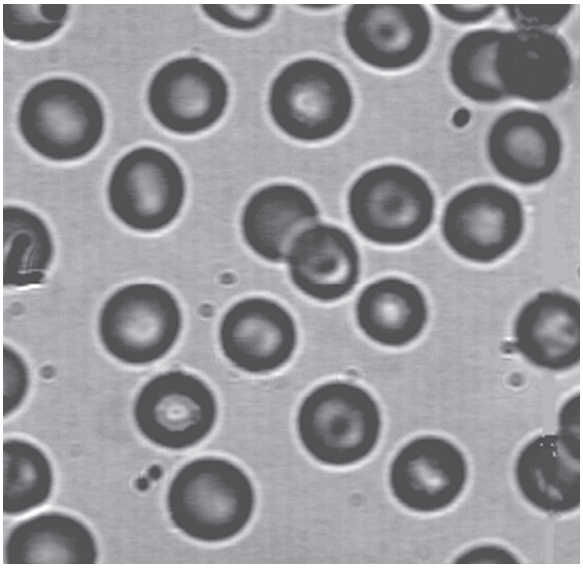
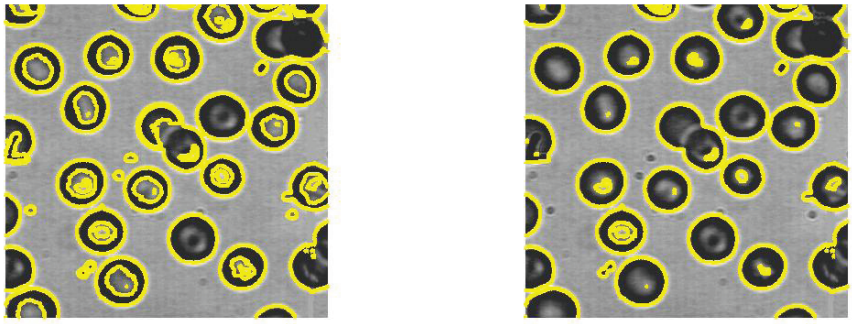


Figure 3.6: The original image.



(a) (b)

Figure 3.7: An example of original image and its MSER results with different parameters for threshold.

by a factor k in scale space. Adjacent Gaussian images are subtracted to produce the DoG images. After each octave, the Gaussian image is downsampled by 2 and the process repeated. In the function, $I(x, y)$ is the image.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

The next step is to detect local extrema in DoG images. Each point is compared with its 8 neighbors in the image and 9 neighbors in the scale above and below. Therefore it is compared with 26 neighbors. It is selected if it is the smaller or larger than all its neighbors.

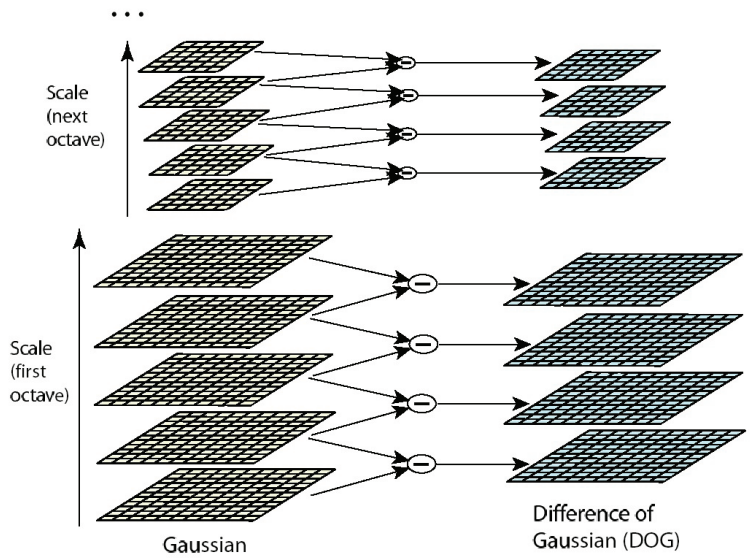


Figure 3.8: For each octave, the original image is repeatedly convolved with Gaussian filters to produce images separated by a factor k in scale space, shown on the left. Adjacent Gaussian images are subtracted to produce the DoG images, shown on the right. After each octave, the Gaussian image is downsampled by 2 and the process repeated.

Once a keypoint has been found, the next step is to perform accurate keypoint localization. Points can be rejected if they have low contrast or are poorly localized along an edge. They use a function to evaluate the extremum to discard unstable extrema with low contrast. They then eliminate edge responses, since DoG has a strong response along edge and some locations along the edge are poorly determined and unstable.

The next step is to assign orientation to the keypoints. They compute image gradients by a simple mask $[-1, 0, 1]$ and get orientation θ for a keypoint.

Then they create SIFT descriptors. Figure 3.9 illustrates the computation of the keypoint descriptor. The coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. The gradient magnitude and orientation at each point in a region around the keypoint are computed and weighted by a Gaussian window. Then the samples are accumulated into orientation histograms over a 4×4 subregion. In Figure 3.9 there are 4 subregions shown on the right. In the experiment, they use 16×16 region, so there are 4×4 subregions, corresponding to 4×4 array of histograms. Each histogram has 8 orientation, so there will be $4 \times 4 \times 8 = 128$ features for each keypoint. Finally, the feature vector is normalized to unit length. Figure 3.10 and Figure 3.11 show an image and its SIFT descriptors.

The applications of SIFT include object recognition [32], robotic mapping and naviga-

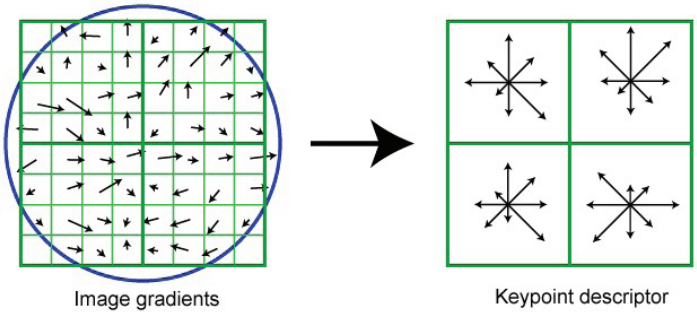
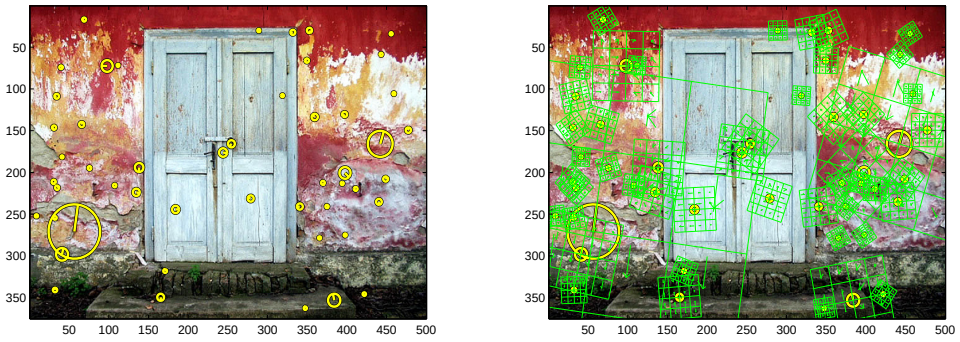


Figure 3.9: The gradient magnitude and orientation at each point in a 8×8 region around the keypoint are computed and weighted by a Gaussian window, shown as the circle on the left image. Then the samples are accumulated into orientation histograms over a 4×4 subregion. In this example there are $2 \times 2 = 4$ subregions shown on the right. The histogram has 8 orientation. There are $2 \times 2 \times 8 = 32$ features in this example. In the experiment they used 16×16 region.

Figure 3.10: The original image.

tion [45], 3D modeling [23], gesture recognition [63], etc. For example, Sivic et al. [47] used SIFT for discovering objects and their location in images. Gordon et al. [20] proposed a system for constructing 3D models from multiple images take with uncalibrated handheld camera and precisely solving for object pose.

In our work, we computed dense SIFT with step 2, i.e. compute a SIFT descriptor every 2 pixels in each row and column. We use the bag-of-words model for feature representation, which is described in Section 3.5.



(a) (b)

Figure 3.11: An example of original image and its SIFT frames and descriptors.

3.4 HOG

Histogram of Oriented Gradients (HOG) are feature descriptors used for the purpose of edge detection. HOG counts the occurrences of gradient orientations in an image. This method was first described by Navneet Dalal and Bill Triggs [8] in 2005. The descriptor is similar to SIFT, but these descriptors are computed on dense grids at a single scale without orientation alignment or overlapping local contrast normalization. Dalal and Triggs performed human detection on the MIT pedestrian database and the INRIA person dataset. They obtained good results on person/non-person classification[8]. Their feature extraction and person detection flowchart is shown in Figure 3.12.

The first step of the chain is gamma/color normalization. They perform gamma equalization on grayscale and RGB color spaces. Then they compute gradients with masks as $[-1, 0, 1]$ and 3×3 Sobel mask. They found that simple 1-D $[-1, 0, 1]$ mask performs best in their experiment. The next step is spatial/orientation binning. An image is uniformly divided into rectangular cells, for example, 8×8 pixel cells. The orientation bins are evenly spaced over 0 to 180 degrees for unsigned gradient. Each pixel within the cell casts a weighted vote for an orientation-based histogram channel. They found that unsigned gradients with 9 histogram channels performed best in their human detection experiments. Choosing the vote weight as the gradient magnitude itself performs the best.

The next step after orientation binning is normalization over descriptor blocks. They group the cells together into larger, spatially connected blocks. The HOG descriptor is then the vector of the components of the cell histograms from all of the block regions. There are two geometries of blocks, R-HOG (rectangular) and C-HOG (circular). A R-HOG example is

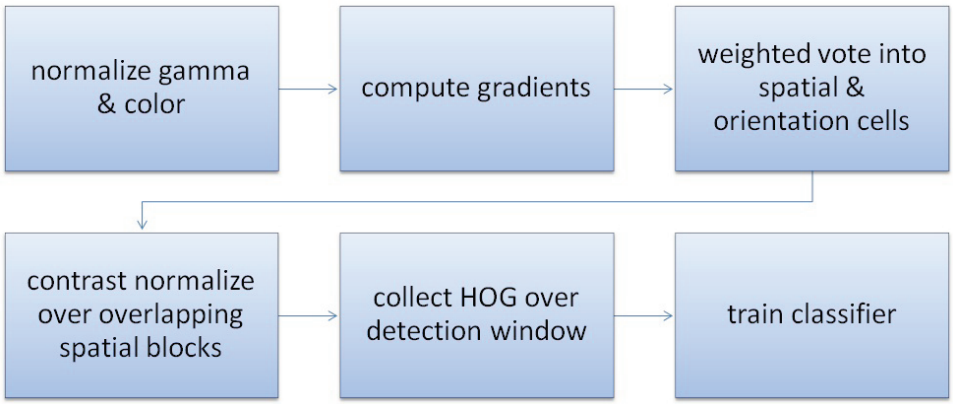


Figure 3.12: An overview of Dalal and Triggs' flowchart of feature extraction and human detection.

shown in Figure 3.13. Figure 3.14 shows a C-HOG geometry. R-HOG is more commonly used than R-HOG. The blocks may overlap, meaning that each cell contributes more than once to the final descriptor. They tried different methods of block normalization. Let vector v contains all the histograms in a block. $\|v\|_k$ be its k-norm and ε is a small constant. The normalization are:

$$\text{L2-norm: } v \rightarrow \frac{v}{\sqrt{\|v\|_2^2+\varepsilon^2}}$$
$$\text{L2-hys: L2-norm followed by clipping}$$
$$\text{L1-norm: } v \rightarrow \frac{v}{(\|v\|_1^2+\varepsilon)}$$
$$\text{L1-sqrt: } v \rightarrow \sqrt{\frac{v}{(\|v\|_1^2+\varepsilon)}}$$

After normalization, they collect normalized histograms from each cell of each block to construct a feature vector. The histogram of one cell is used more than once, but with different normalization. If the image size is 64×128 and suppose they use 8×8 pixel cell and 16×16 pixel block, a block is formed by 4 cells. If using 9 orientation bins, there are 36 features in one block. Block spacing stride is set to 8 pixels, hence there are 15×7 blocks and the total number of feature for an image is $15 \times 7 \times 36 = 3780$. Then they perform classification using linear SVM. In fact, Dalal and Triggs [8] proposed that using 6×6 pixel cells and 3×3 cell blocks gives the best result.

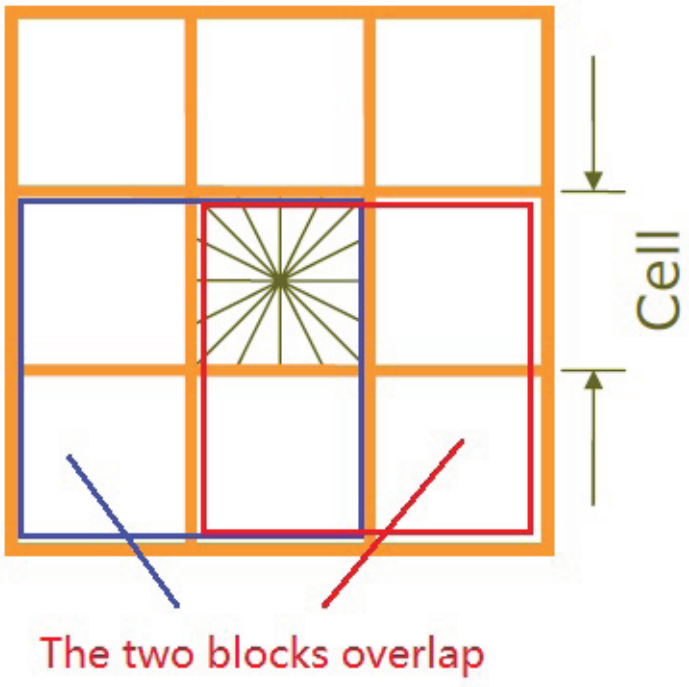


Figure 3.13: The two blocks overlap. In this case, a block contains 4 cells and the histogram of each cell is used in 4 blocks.

HOG is often used for human detection and it was shown that HOG descriptors outperform other features for pedestrian detection[8]. In 2006 HOG was applied for detecting moving people in film or videos [9]. It can be used for object detection as well [13].

An example of HOG features is shown in Figure 3.15 and Figure 3.16. In this example, HOG feature are computed in each cell with size of 8×8 and 16×16. The results are the pictorial rendition of the features. The dominant orientation of some cells is obvious to see, although it is not used as feature. The results shown below used number of orientations as 9, which is proved to have the best performance in human detection problems.

In our work we filter the intensity data using filter kernel Sobel operator and create orientation bins. The signed orientation of 0 to 360 degrees are divided into n histogram channels. We did not use cells but uniform layers. The combination of the histograms in layers is then used as a feature descriptor. We will discuss the detail of our way to construct HOG feature in Chapter 4.

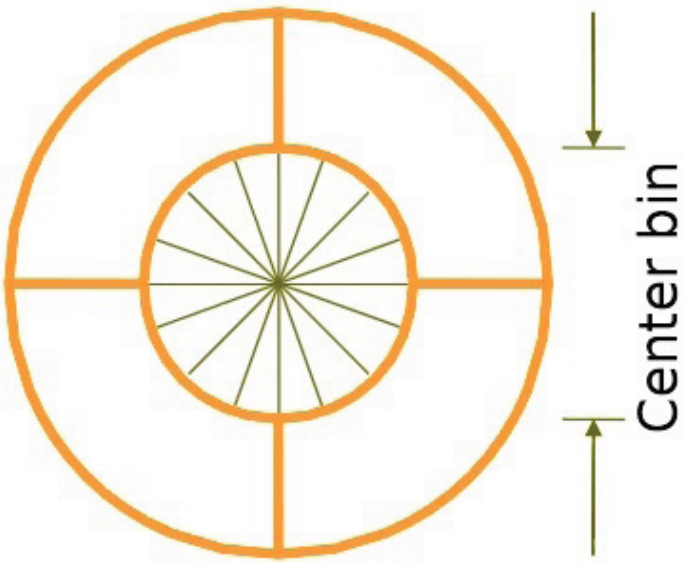


Figure 3.14: A C-HOG geometry.

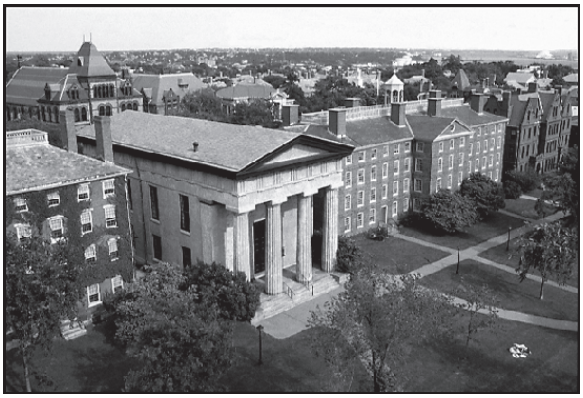
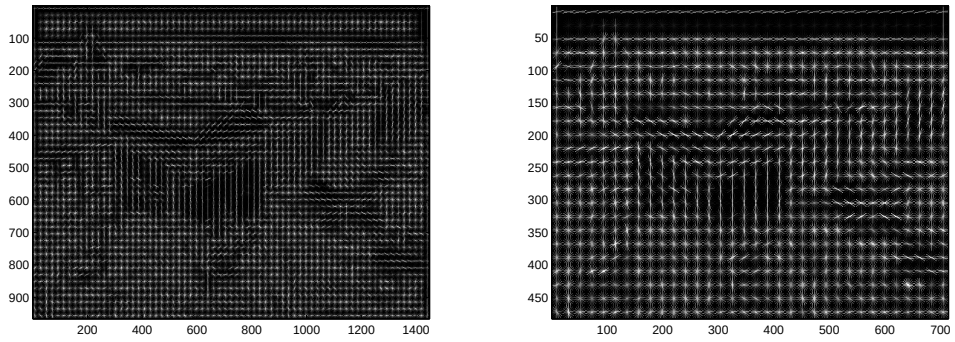


Figure 3.15: The original image.

3.5 Feature Representation

For different computer vision problems, different image features are appropriate. In this section we will talk about feature representation methods.



(a) (b)

Figure 3.16: An example of original image and its HOG features with different cell sizes. The number of histogram channels is 9. (a) is HOG features with cell size 8 and (b) is with cell size 16.

3.5.1 Feature Vector

In machine learning, a feature vector is an n-dimensional vector of numerical features that represent an object. When representing images, in the simplest case, the n-dimensional vector can just correspond to the n intensity values of the image, one value per image pixel, piling pixel intensities in row-wise fashion into the image vector. However, most often, higher-level features are necessary to obtain a satisfying performance. These higher level features are computed from pixel intensities, but usually one higher level feature will depend on a few pixel intensities from some small image patch. Higher-level features can encode more interesting image properties, such as certain textures and patterns. The vector space corresponding to the feature vectors is called the feature space. Thus each image is represented as a point in d-dimensional space. Sometimes dimensionality reduction methods are needed when the dimension of feature space is high.

3.5.2 Bag-of-Words Model

The bag-of-words (BoW) model is a simplifying representation first used in natural language processing and information retrieval [22]. In this model, a text is represented as an unordered collection of words. It is commonly used in document classification, where the occurrence of words is a feature of the text. A vocabulary or dictionary should be constructed first and the text can be convert to a vector representing the occurrence of words in dictionary. Figure 3.17 shows a bag-of-words model, in which the dictionary is “John”: 1, “likes”: 2, “to”: 3,

"watch": 4, "movies": 5, "also": 6, "football": 7, "games": 8, "Mary": 9, "too": 10. This kind of representation can be used for email filtering. Recently, bag-of-words model has become very popular in computer vision [48]. Figure 3.18 shows a toy example of how vocabulary of patches can be used in image categorization.

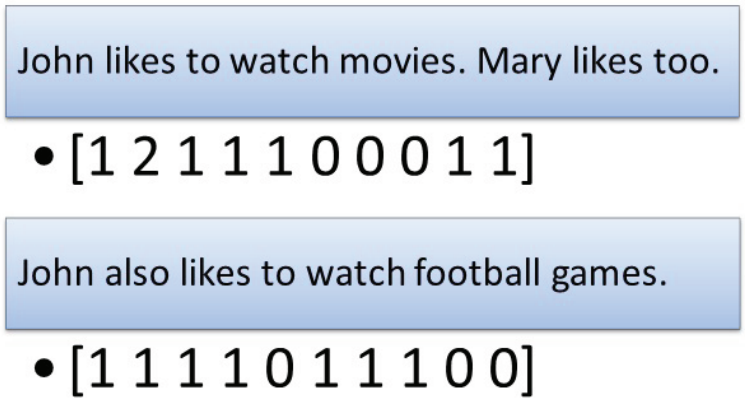


Figure 3.17: An example of bag-of-words model.

In a bag-of-words model in computer vision, an occurrence of a particular image feature is treated as occurrence of some word. However, the dimensionality of typical useful image features is huge. For example, SIFT feature have dimensionality of 128. Therefore, we cannot treat each particular SIFT feature as a separate word. There are many SIFT descriptors that are slightly different, but arise from very similar looking image patches. We want to treat all similar SIFT descriptors as an occurrence of the same word. Therefore, similar SIFT descriptors (or similar image features, if using other features), are mapped to the same "codeword" and treated as occurrence of the same "word", or "codeword". This step is really a quantization step, when multiple range of feature values is converted to a much more narrow range.

Thus, to achieve image representation on the BoW model, there are three steps: feature detection, feature representation and codebook generation[12]. After feature detection, each image pixel is abstracted by some descriptor, which is usually a vector or a patch. For example, when using SIFT features, each image pixel is abstracted by a vector (SIFT descriptor) which has dimension 128. The next step of BoW model is to convert these descriptors to "codewords". One simple method of obtaining codewords is to apply k-means clustering over all the descriptor vectors. The cluster centers become codewords, which are also called visual words or representative features. The number of clusters is the number of words in the dictionary. Therefore each descriptor (a patch or a feature vector) is mapped to a codeword and each

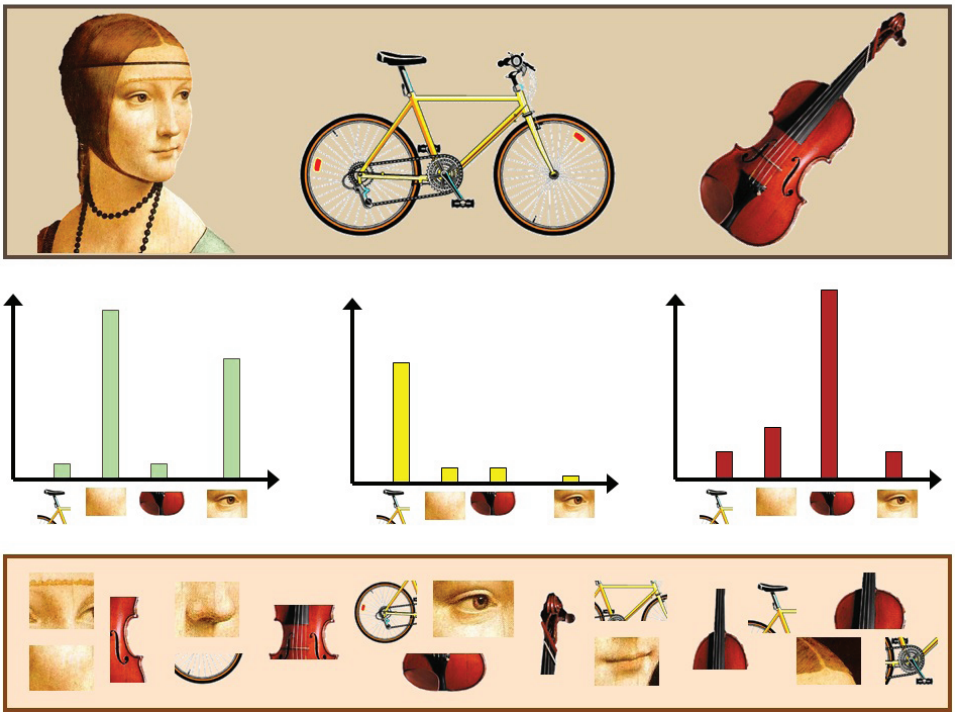


Figure 3.18: A toy example of how bog-of-words is used in image categorization. The vocabulary is constructed with patches. The images can be classified by the histogram of visual words.

image can be represented by the histogram of codewords.

3.5.3 Spatial Pyramid Representation

The bag-of-words model has one significant drawback for image analysis. It completely ignores the spatial information in the image. That is, it counts visual word occurrences, but does not care where in the image these visual words came from. However, there is a lot of information in the spatial domain in an image.

To address this problem with the bag-of-words representation, that is, to help to include some spatial information of the image into the feature representation, Lazebnik et al.[29] proposed spatial pyramid representation. It is based on the Bag-of-Words model. Spatial pyramid representation works by partitioning the image into sub-regions and computing histograms of codewords inside each sub-region. They built a pyramid in the image space at different levels as shown in Figure 3.19.

These histograms from different scales and image sub-regions are piled together into one

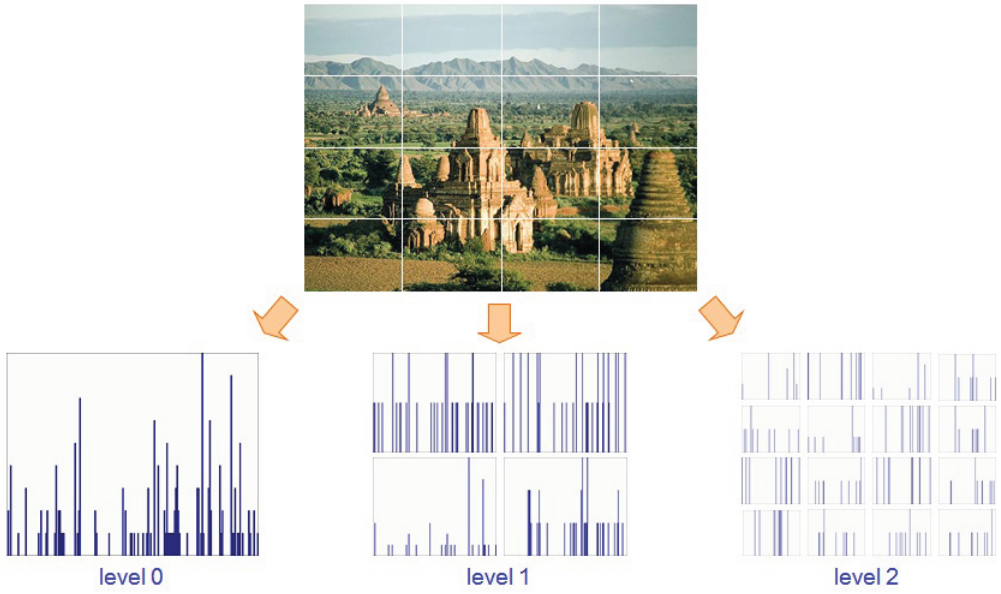


Figure 3.19: An example of how an image is partitioned into sub-regions at different levels. Histograms of codewords are computed on each level. For different levels, the size of sub-regions are different.

feature vector. In this way, the feature vector will have a particular representation for the upper right part of the image, upper-right part of the image, center part of the image, etc. So if there is something in particular in the upper left part of the image that can help to recognize an image class (like, say, texture features corresponding to the sky in the upper left region), this pyramid feature representation will be able to supply this information to the classifier, since we have a particular part of the feature vector corresponding specifically to the upper left part of the image.

Lazebnik et al. [29] extracted feature descriptors from an image and construct visual word dictionary using k-means. They built spatial histograms and train SVM on Caltech 101 dataset. They show improved results over the bag-of-words representation for recognizing natural scene categories.

In our work, we perform pyramid representation in a different way. Instead of partitioning images into boxes, we divide them into layers since it appears that there is no useful information in horizontal direction. The details will be described in Section 4.2.

Chapter 4

Epilepsy Lesion Classification

In this chapter, we will discuss the work we have done in detail. First we will give an introduction to our epilepsy classification work. We will talk about our goals. Afterwards we will introduce the datasets we used in our work. The data includes sliced profile images with different locations and different labels. For training, the samples with epilepsy lesion needed to be classified by a human expert to find the disorders in neuron layers. Then we will talk about our approach of classifying this data. We tested different feature types commonly used for object detection and classification, to find those that perform best for our problem. The classifiers we tested include nearest neighbor, boosting and SVM. We will compare the classification results using different features.

4.1 Introduction to the Dataset

Epilepsy is a set of chronic neurological disorders characterized by seizures. Some definitions of epilepsy require only a single seizure combined with brain alterations which increase the chance of future seizures. Factors like brain trauma, strokes, brain cancer, and drug and alcohol misuse may be associated with the causes of seizures. Focal Cortical Dysplasia refers to the malformations in cortex and it is the most common pathology in MRI-negative epilepsy. Type 1 FCD is related to radial abnormalities or tangential abnormalities. Quantifying Type 1 FCD can have significant impact on histopathology correlation studies. The datasets we used contain three patients' sliced profile images with different locations and labels.

4.1.1 CorticalProfiles_NeuN

In the CorticalProfiles_NeuN dataset, there are samples from two patients with epilepsy, numbered as EPI_P006 and EPI_P014. The original image of profile is with high resolution and the neurons are clearly visible as brown blobs. The size of profile images can be as big as 10000×20000. Figure 4.1 is a profile example of EPI_P014. It shows the full map of one brain profile. Obviously we cannot use the whole profile images directly, so samples are taken from the boundary of the profile. The information we are interested in is in the cortex. The blue lines show the positions of the sampling. The blue lines are made perpendicular to the profile boundary. The sampling positions are uniformly distributed along the cortex. In this set of data, the sampling is dense around the boundary. In this dataset, EPI_P014 has 5 profiles like that taken from different angles. EPI_P006 has 2 profiles. Figure 4.2 shows the local detail of the profile. The neurons are visible as brown blobs. Figure 4.3 shows two sliced samples from the whole profile. Each sample is a rectangle sliced along the direction of the blue line. The width of the images are made identical among all the samples, which is 500.

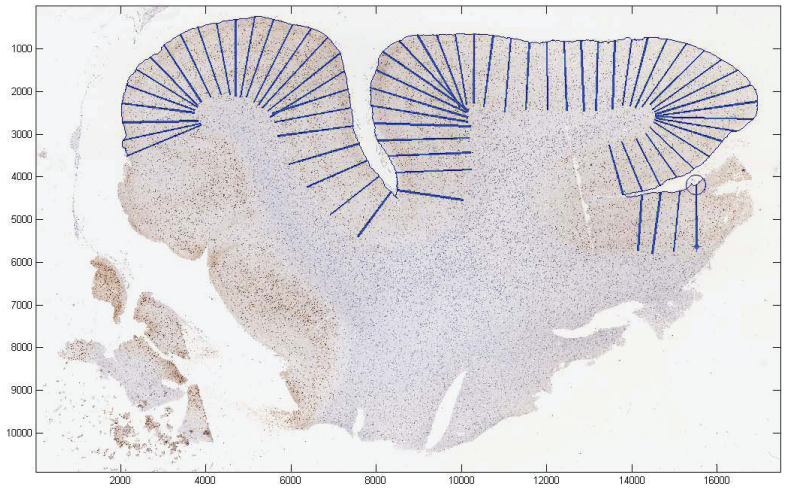


Figure 4.1: A brain histology image of EPI_P014. The original image is a high definition image. The blue lines indicate the positions of sampling.

In the CorticalProfiles_NeuN dataset, we have the sliced profiles of EPI_P006 and EPI_P014.

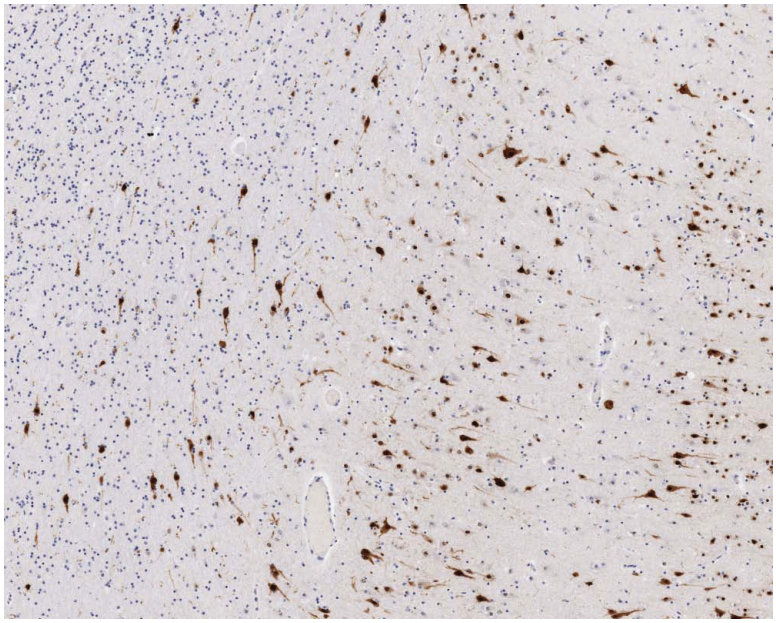
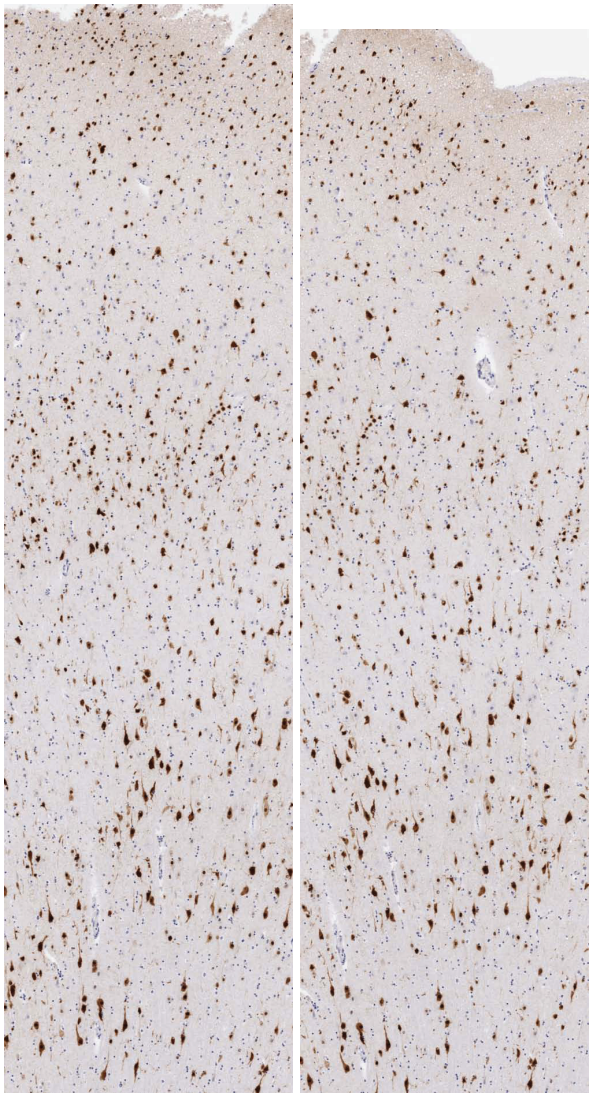


Figure 4.2: This is a small part of the whole histology image of EPI_P014. The bigger brown blobs indicate the neurons.

Some of the samples of EPI_P014 show abnormalities. The samples of EPI_P014 are labelled as normal or abnormal. For the 194 samples of EPI_P006, the pathologists cannot identify any abnormalities. There are 324 samples in EPI_P014, 67 of them are abnormal that correspond to epilepsy lesion. The other 257 samples are normal.

4.1.2 corticalQuantHist

The other dataset we used is the corticalQuantHist dataset. Like before, samples of EPI_P006 and EPI_P014 are included, yet we have another patient EPI_P015 in this dataset. The sliced images are similar to that in the CorticalProfiles_NeuN dataset. The main difference of this dataset and the previous is that in this dataset, for each sliced histology image, the pathologist has labeled its layering. Each sample is divided into 6 horizontal layers. The boundaries between neighboring layers are given to us. The disorders in layers are mostly likely to appear in layer 3 and layer 5 because these are usually the thickest layers. We are given the count of neurons in layer 3 and layer 5 for nearly every sliced profile. Another difference is that in this dataset, the sampling on the whole profile is less dense than before. Instead of sampling uniformly along the cortical surface, the medical imaging scientists sampled on some contours along the boundaries of profile. On each contour there are a few sliced profiles. Figure 4.4



(a) (b)

Figure 4.3: Two sliced profiles of EPI_P014. Each sample corresponds to one blue line in Figure 4.1. We performed the classification on this kind of images.

and Figure 4.5 show the profile with the neuron densities of layer 3 and layer 5 marked. The profiles with layering information are shown. The brighter the dot is, the denser the neurons is in the corresponding sample.

The count of neurons in layer 3 and layer 5 are computed manually by a human expert in the middle section of each layer. The size of the inner section is 201×201. In Figure 4.6, the layer 3 of one profile image is shown along with the inner section marked. The number of neurons in this example is 18.

From Figure 4.7 it is easy to notice that the 6 layers have different sizes. Layer 3 and layer

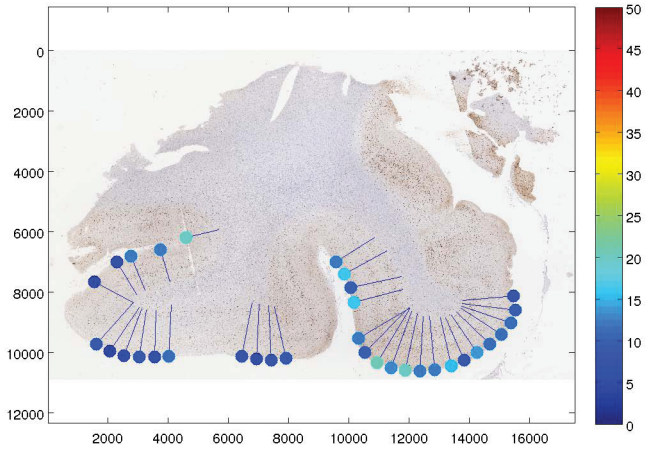


Figure 4.4: A profile of EPI_P014 with sampling positions and neuron densities in layer 3 marked.

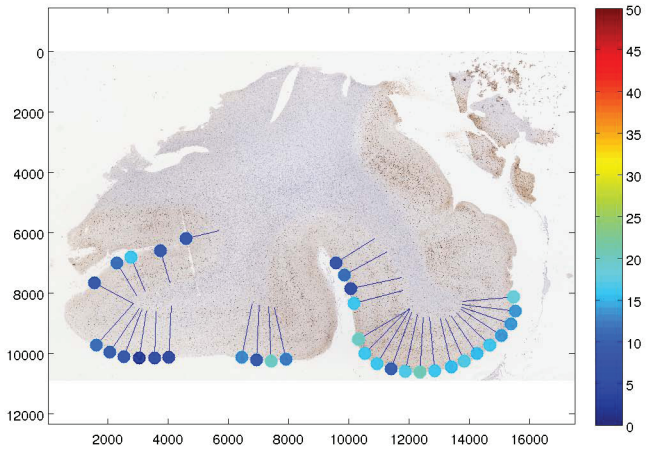


Figure 4.5: A profile of EPI_P014 with sampling positions and neuron densities in layer 5 marked.

5 are usually thickest. Also, the actual orientations of boundaries between layers and profile boundary are not always horizontal. The manually labeled boundaries are not parallel to the actual boundaries. This may cause inaccuracy in counting neurons.

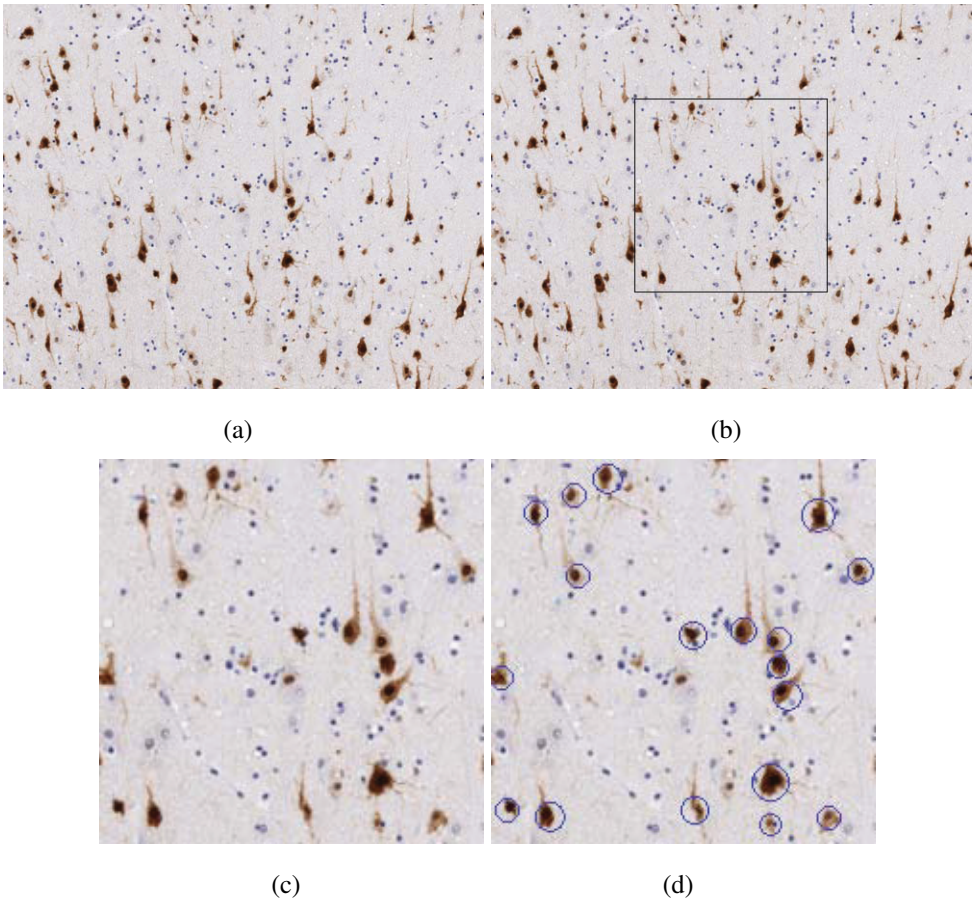
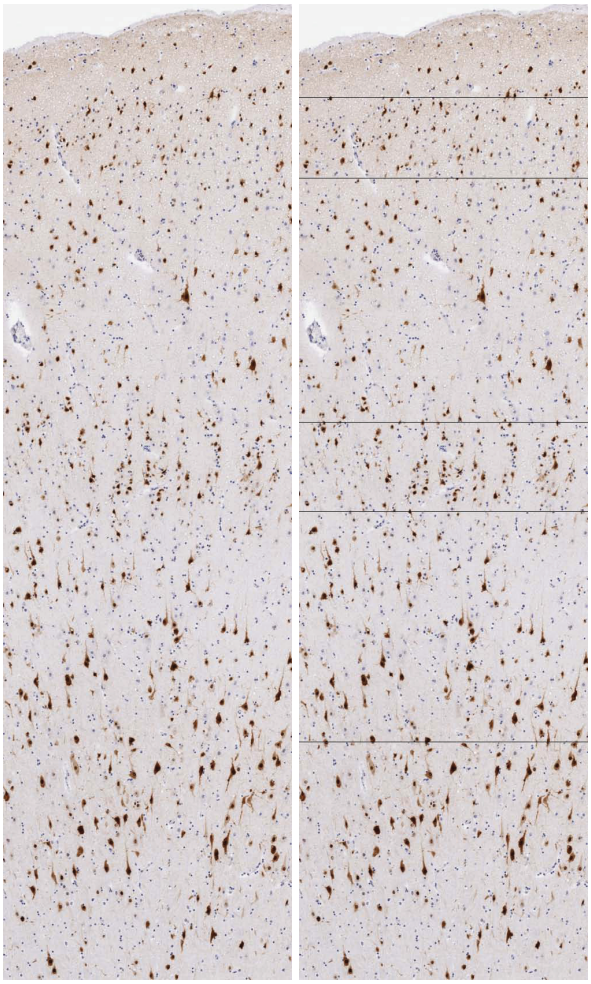


Figure 4.6: In (a) there is the layer 3 of one profile image. The inner section where the count of neurons was going on is marked in (b). In this layer 3 image, the count of neurons in its inner section is 18, as shown in (d).

All the sliced profile images have width of 500. They have different heights. We have the normal/lesion labels of EPI_P014, in which 67 samples are abnormal and 184 are normal. The 52 samples of EPI_P006 and EPI_P015 do not show any abnormalities.

4.1.3 H&E&NEUN stain

The samples in this dataset are made with H&E&NEUN stain. This kind of samples are easier to obtain. The character of this dataset is that the samples are stained and the neurons are difficult to see. Figure 4.8 shows two samples in H&E&NEUN stain dataset. In some images, there are significant noise artifacts. This might affect the classification result. This dataset contains the stained profiles of EPI_P014. The samples share the same labels with their correlative image in corticalQuantHist dataset.



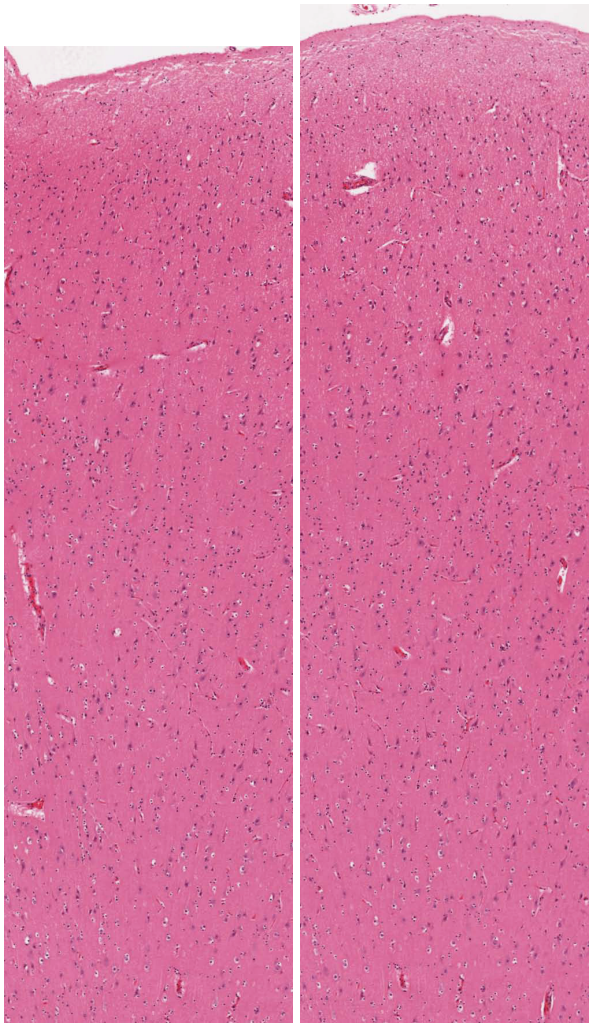
(a) (b)

Figure 4.7: A sliced profiles of EPI_P014 and its layer boundaries.

4.2 Feature Extraction

In this section we will talk about the features we used for the classification problem. We evaluated several different feature types that were successful for a variety of applications in computer vision, in order to find features that are most appropriate for our problem. We will discuss in detail about how we extracted SIFT, HOG, histogram of color, histogram of intensity and stable regions features. Also, we will talk about the feature representation method we used.

First of all, we want to talk about how and where the features are extracted and generated. We know that the FCD we are trying to find appears as disorder in layer, so we decide to divide an image into layers and extract features from each layer. The neurons are grouped into 6 different layers and we have ground truth of cortical layering for some data. We did not use the ground truth of layering, but we used uniform layers instead, inspired by Lazebnik et al.'s



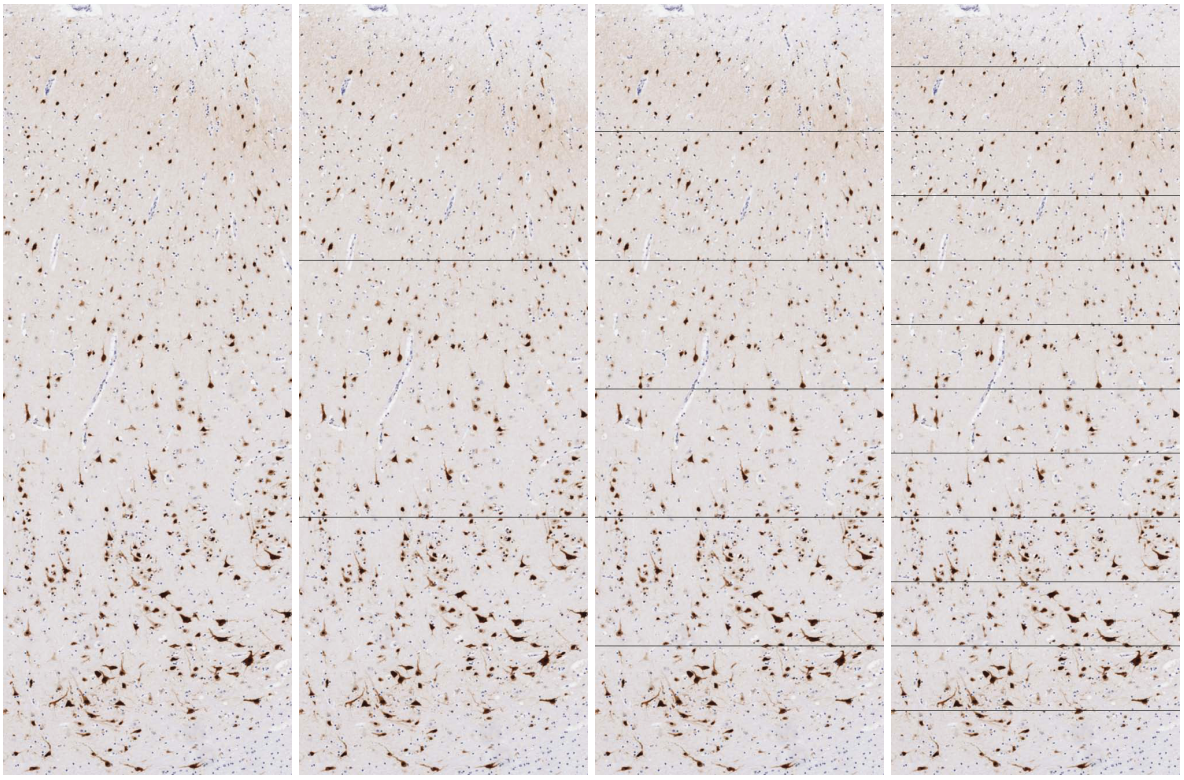
(a) (b)

Figure 4.8: Two stained profiles of EPI_P014.

work [29]. Since we do not know what is the true underlaying layering of the image, we divide an image into different set of layers and use those for feature extraction. In particular, we break image in 1, 3, 6, 12 layers of equal height, and combine information extracted from each of these layering together. Figure 4.9 shows how we divided the image. Features are extracted from each layer and combined together to represent the whole image.

4.2.1 SIFT Extraction

As we mentioned before, SIFT is a widely used algorithm for detecting and describing features. In our work we extracted dense SIFT feature from the image. Experimental evaluations show that for object or scene classification problems, better results are often obtained by com-



(a) (b) (c) (d)

Figure 4.9: We divided the sample image into uniform layers. The number of layers are 1, 3, 6 and 12.

puting dense SIFT features [29]. Instead of computing SIFT descriptor just at sparse interest points, we compute a large number of descriptors over dense grids. Dense SIFT descriptors provide more information than sparse descriptors, which just focus on a small set of interest points.

We use VLFeat open source library [56] and MATLAB environment to extract dense SIFT features. We used the vl_dsift command to compute dense SIFT features. The images are converted into grayscale to compute SIFT. To reduce the computational complexity, we extracted SIFT features every 2 pixels from the whole image. An image will have about 200,000 SIFT descriptors. Each descriptor has a dimension of 128. It is obviously too much computation for classification if we use all the descriptors.

We partition the descriptor space into informative regions. As in Sivic et al.'s work [47], we try to find "visual words", that is image features that occur relatively frequently and provide rich description of a small image region. Raw SIFT features are not necessarily useful for this purpose, since they may exhibit significant variation for small change in image appearance. Therefore we take an approach in Sivic et al.'s work [47] and cluster SIFT features into repre-

sentatives that may be more insensitive to image region transformations. We collected the SIFT features of all images from training set together and use k-means to cluster those features into k clusters. Each SIFT descriptor is a 128-dimensional vector. The centers of these k clusters become the cluster representatives.

K-means clustering is a method of cluster analysis. It was first proposed in 1967 by James MacQueen [34]. The goal of k-means is to partition n data into k clusters and minimize the within-cluster sum of squares. Suppose there are n samples $(x_1, x_2, ..., x_n)$, each sample is a d-dimensional vector. K-means aims to partition the n samples into k sets $S_1, S_2, ..., S_k$ to minimize the within-cluster sum of squares $\sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2$. As shown in Figure 4.10, k-means is an iterative clustering algorithm. The algorithm initializes k cluster centers randomly and assign each sample to the closest cluster. Then the following two steps are iterated. The first step is the update step. In this step we calculate the new means of the k clusters to be the centroids of all points currently assigned to the cluster. The second step is the assignment step. In this step, each sample is assigned to the cluster whose mean is closest to the sample. The iterations stop when the procedure has converged, that is, when the assignments do not change any more. It can be shown that the update and assignment step lower the value of the objective function, thus the iterative application of these steps lowers the objective function. K-means minimizes the within-cluster sum of squared differences between samples and cluster means. The algorithm finds only a local minimum, but global optimization is NP-hard. What is good about the algorithm is that at each iteration step, many samples potentially change their cluster assignment, while the objective function still goes down. Therefore convergence is relatively rapid, usually.

Figure 4.11 shows an example of k-means clustering. There are 200 2-dimensional data points and the number of clusters is set to 2. In the first iteration shown in Figure 4.11(a), two means are initialized and all the samples are assigned to a cluster, marked as red or blue. Then in the Figure 4.11(b), the means of the 2 clusters are updated and each sample is reassigned to the closest mean. We can see that the sum of squares drops with more iterations. The algorithm converges after 4 iterations.

In our work, we used the vl_kmeans function in the VLFeat open source library [56]. We clustered the SIFT features from the training set into k clusters. Each of the k cluster centers becomes a cluster representative. We want to quantize all the SIFT features into just k bins, each bin corresponding to a cluster representative. This will group similar SIFT features together, so that they are treated as the same feature. Each cluster of SIFT features will correspond to a set of similar image patches, and the distribution of these patches will describe texture patterns

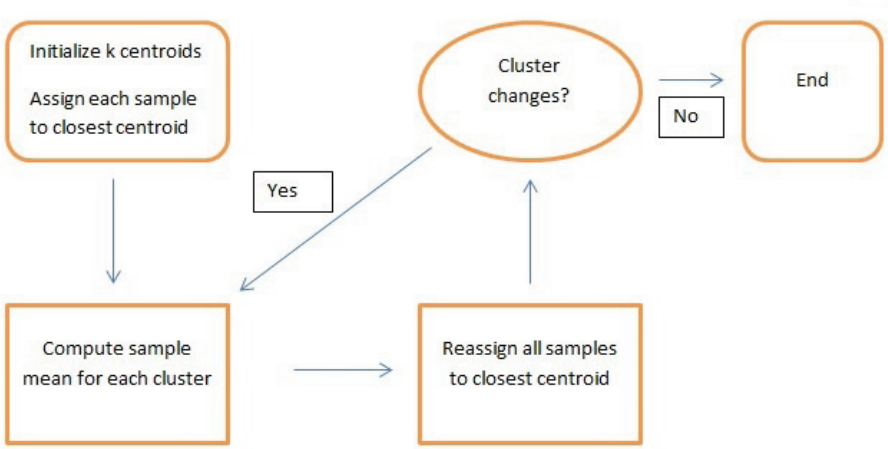


Figure 4.10: This is the k-means algorithm procedure.

in the image. We therefore assign each SIFT feature to its closest cluster representative. Let cluster representatives be indexed by $1, 2, ..., k$. Each SIFT feature is converted to an index in set $\{1, 2, ..., k\}$ using this procedure. Now we compute the histogram of these representative features, that is histogram that has k bins, for each layer of each image. Figure 4.12 shows the procedure of obtaining representative features and histograms. Figure 4.13 shows a histogram of representative features in one layer. The number of representative features is $k = 100$.

For each layer of an image, every SIFT descriptor is assigned to a cluster. We try several layerings, namely first image is not broken in any layers, then it is broken in 3 layers, then in 6, 12 layers, as shown in Figure 4.9. One layer means using the whole image. We obtain histograms of representative features for each layer, as shown in Figure 4.14. Therefore we have to use the information of $1 + 3 + 6 + 12 = 22$ layers to represent one image. Since we do not know what is the optimal number k of representative features to use, we tried k =50, 100, 200 and 500. For each value of k we compute the histogram and pile these histograms into one feature vector. This means there are four histograms for each layer. We have $50 + 100 + 200 + 500 = 850$ features to store the four histograms. We combined all the histograms from each layer together to make the feature vector. Therefore for one image there will be $850 \times 22 = 18700$ SIFT features. Also, we applied different bin size of dense SIFT descriptors, which corresponds to different SIFT keypoint scales. We set bin size as 5 and 8 as well as the default value 3. This will triple the number of features to $3 \times 18700 = 56100$. Figure 4.15 shows all the SIFT features detected in one image. We extracted histograms of representative features from the whole dataset, both training set and testing set. Figure 4.16 shows the overview of

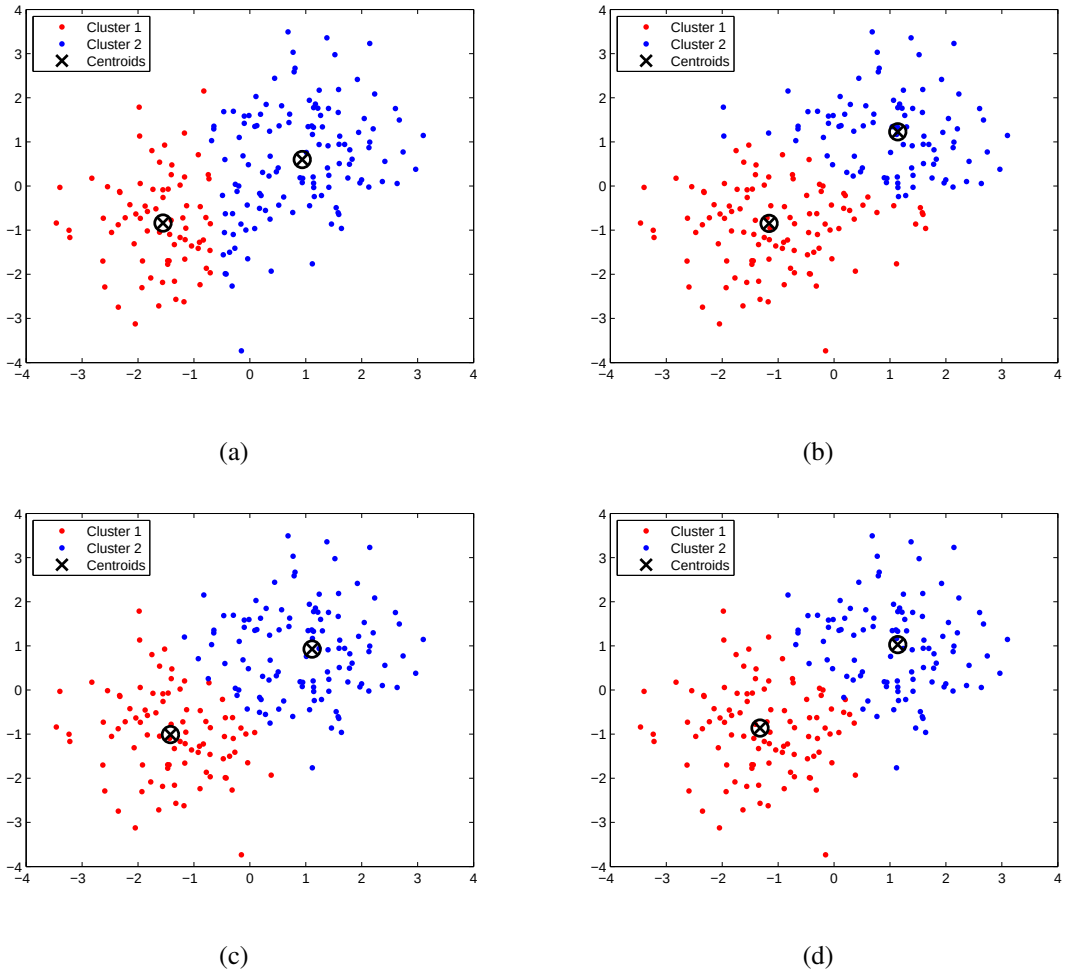


Figure 4.11: An example of the k-means clustering of 200 2-dimensional data. We set number of clusters as 2. The red and blue dots indicate two clusters. The algorithm converges after 4 iterations. The total sum of distances decrease after each iteration.

SIFT feature extracting procedure.

With the same idea, we used the histogram of representative features on color as well. We clustered the RGB colors into 3, 5 and 7 bins using k-means and the color cluster center became representative features. Histograms of each layer are computed and combined together. There are 15 color features in one layer. We also break the image into 1, 3, 6 and 12 layers as shown in Figure 4.14. Therefore we have 330 color features for each image.

4.2.2 HOG Extraction

In our work, we extracted HOG features. Although HOG is most widely used for pedestrian detection problem, it works for a variety of other cases as well [62]. We performed our

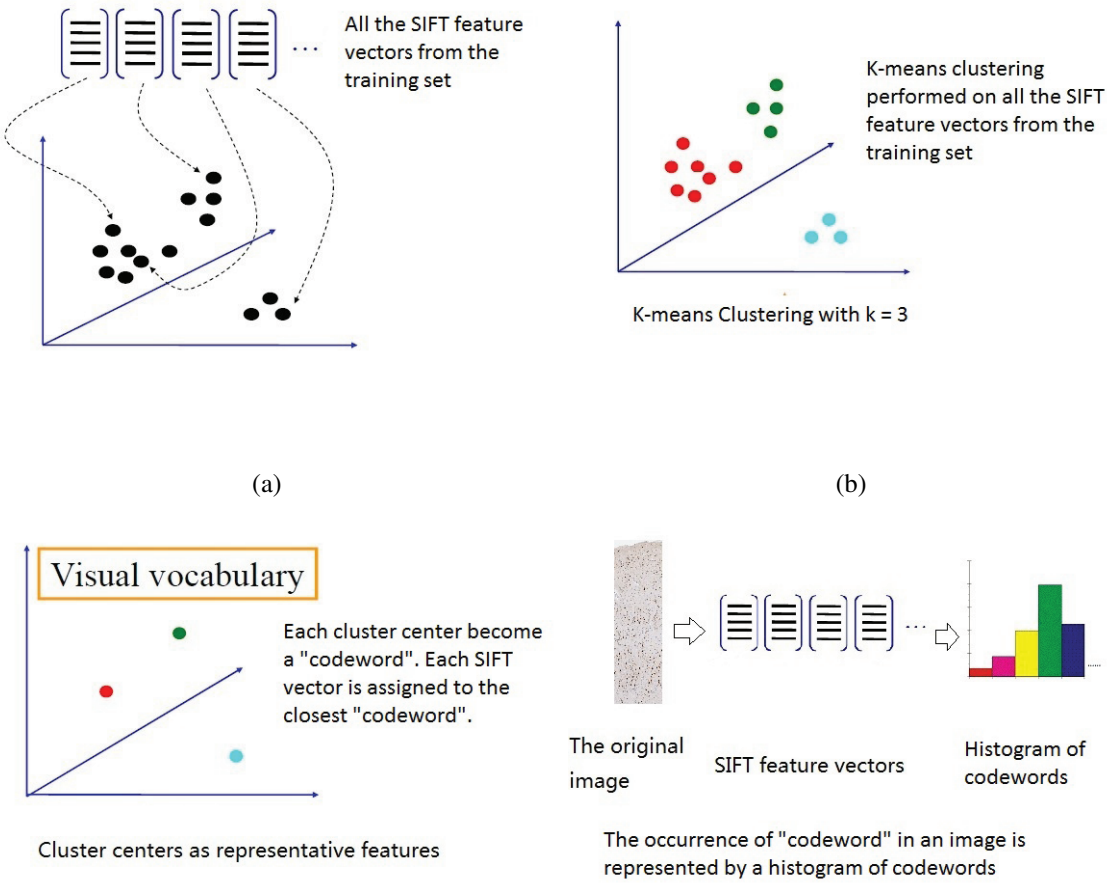


Figure 4.12: The procedure of obtaining codewords and histograms. In (a), all the SIFT feature vectors from the training set. Each SIFT descriptor has a dimension of 128. In (b), k-means clustering is performed on all the SIFT descriptors in training set. In (c), each cluster center become a "codeword". Each SIFT vector is assigned to the closest "codeword". The occurrence of "codeword" in an image is represented by a histogram of codewords, as shown in (d).

extraction in a slightly different way with Dalal and Triggs's method [8]. First, we computed intensity gradients for each pixel of the image. We used Sobel operator to perform it. For horizontal and vertical directions, a Sobel operator is a simple 3 by 3 matrix that is convolved with the original image to calculate approximations of the derivatives. If A is our image, the G_x and G_y are computed as the convolution of Sobel operator and A. The gradient magnitude is $\sqrt{{G_x}^2 + {G_y}^2}$. The orientation can be computed as arctan $\frac{G_y}{G_x}$. The gradient is signed so there are 360 degrees of orientations. We set a threshold to exclude pixels with smaller gradient

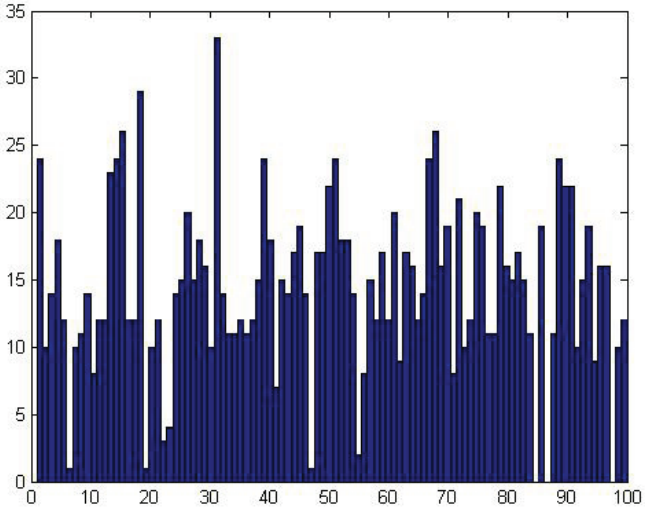


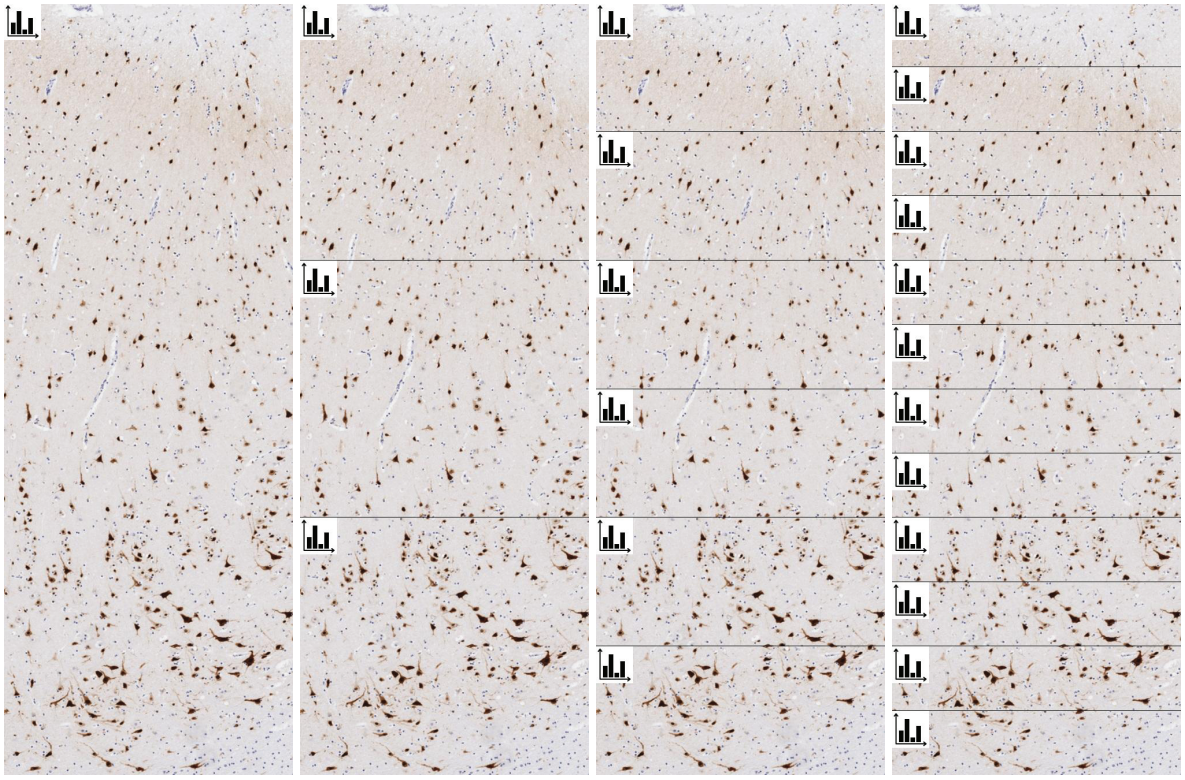
Figure 4.13: This is an example of the histogram of representative features in one layer. The number of representative features is 100.

magnitude, in order to get rid of noisy pixels. We sorted the gradient magnitudes and exclude 20% of pixels with the least magnitude.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \, , \, G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A$$

Then we performed orientation binning. The 360 degrees of orientation are divided into n bins. In Dalal and Triggs's experiments [8] of human detection, they proposed that using 9-bin histograms gives the best performance in their problem. They did not use orientations of signed gradients so there are 180 degrees of orientations. In their work, they had images of size 64×128 to detect pedestrians. They divided the image into pixel cells and cell blocks and performed normalization on overlapping blocks. They compute the HOG descriptors in cells and blocks. They combined the HOG descriptors in the blocks to form a feature vector for an image. They used HOG features for human detection. Figure 4.17 shows the image they used for human detection and its HOG descriptors.

We face a different situation in our problem. In our work, the images we used are big and the distribution of neurons are nearly random. It is hard to find useful information in local regions. The neurons are located in different positions in the images and it does not make sense to compute local histograms in small cells and combine them to a feature vector. The



(a) (b) (c) (d)

Figure 4.14: We compute histogram of representative features in each layer. One little icon means there is a histogram in the layer. There are 22 histograms representing one image.

SIFT bin 3	• number of layers: 1, 3, 6, 12 • number of representative features: 50, 100, 200, 500 • total number of features: 18700
SIFT bin 5	• number of layers: 1, 3, 6, 12 • number of representative features: 50, 100, 200, 500 • total number of features: 18700
SIFT bin 8	• number of layers: 1, 3, 6, 12 • number of representative features: 50, 100, 200, 500 • total number of features: 18700

Figure 4.15: All the SIFT features detected. The total number of SIFT features is 56100.

difference among image sizes can also make the dimensions of HOG feature vectors different. Figure 4.18 shows what it looks like if we compute HOG descriptors in every 8×8 cell.

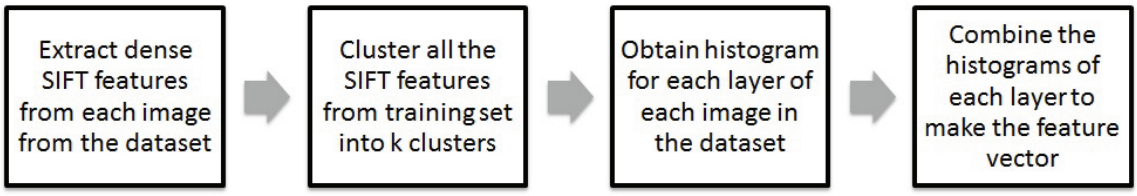
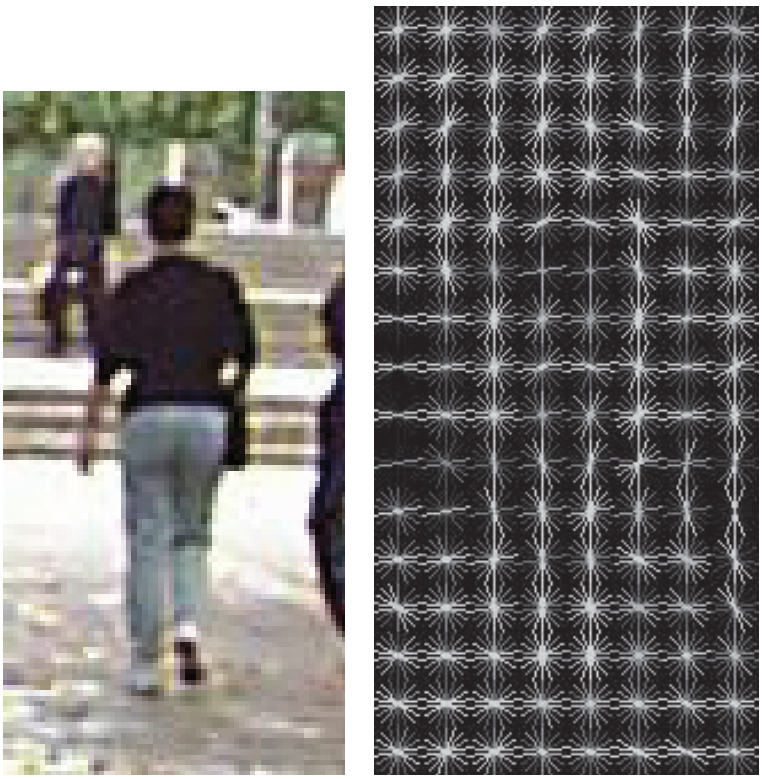


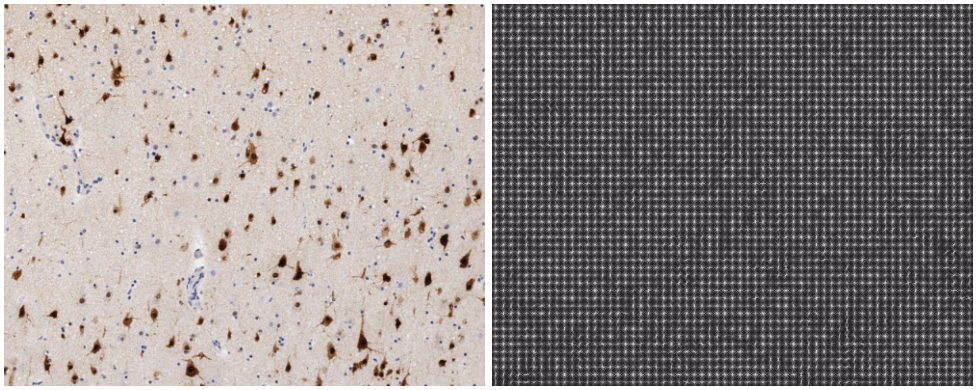
Figure 4.16: The overview of SIFT feature extracting procedure.



(a) (b)

Figure 4.17: A sample image of a pedestrian and its HOG descriptors.

In our work, we did not compute histograms in small cells. Instead we compute histograms from layers since the FCD we are interested in are related to disorder in layers. The layer partition we used are different from the SIFT extraction case. The number of layers are set



(a) (b)

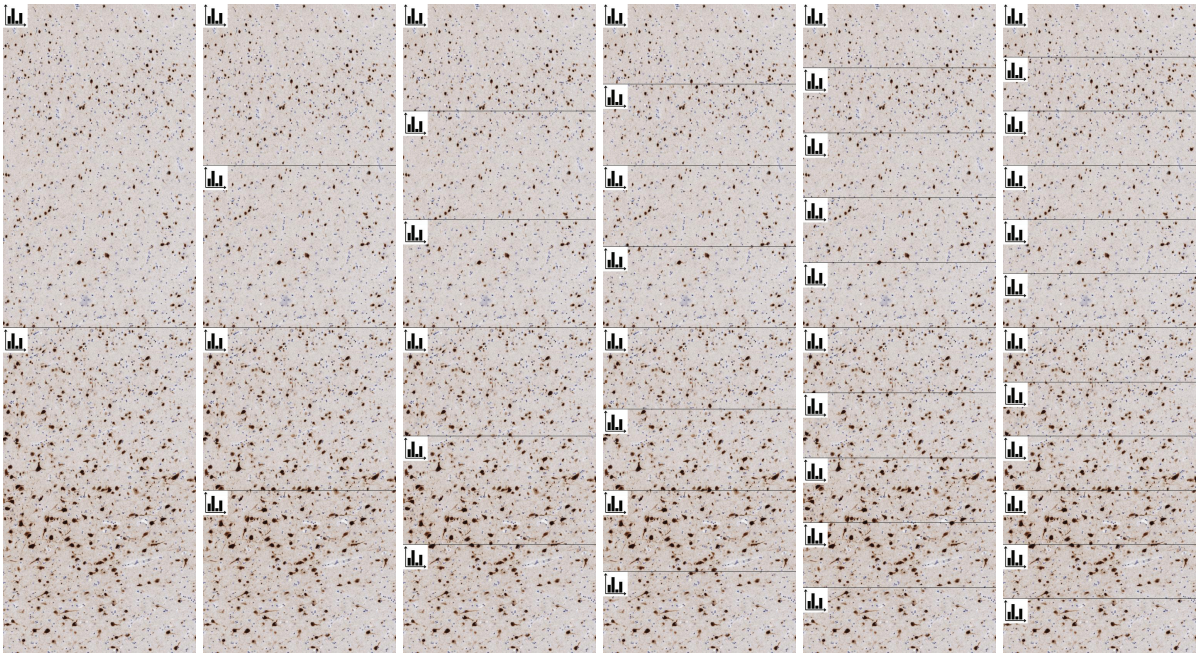
Figure 4.18: A part of sample image in our dataset and its HOG descriptors. The cell size is set to 8.

to 2, 4, 6, 8, 10 and 12, as shown in Figure 4.19. The histogram channels are evenly spread over 0 to 360 degrees. In each layer, we count occurrences of gradient orientations and assign them into histogram channels. We used number of bins as 4, 8 and 16. Therefore we need 28 features to represent the information detected in one layer. In the end, the histograms of all the layers and all the bins are combined to be a HOG feature vector. There are 1176 HOG features in one image in total.

4.2.3 Histogram of Normalized Intensity

Histogram of normalized intensity is another type of features we tried. We decide to use this feature because we observed that there are some difference in the intensity distribution between normal and abnormal samples. For example, some abnormal samples have very sparse distribution of neurons, and since neurons are darker than the background, those abnormal samples could be distinguished from normal samples based just on the intensity histograms. We convert the images into grayscale and normalized them. Then we divide the intensity values into uniform channels. We computed histograms with 2, 3, 4, 5 and 6 bins, where intensities were partitioned uniformly into the corresponding number of bins. Figure 4.20 and Figure 4.21 show the intensity map of a normal sample and an abnormal sample. Some difference can be seen.

Histograms of intensity are obtained from all the layers shown in Figure 4.14 and combined together as well. We have $2 + 3 + 4 + 5 + 6 = 20$ features in one layer. The number of HOG features in one image is $20 \times 22 = 440$.



(a) (b) (c) (d) (e) (f)

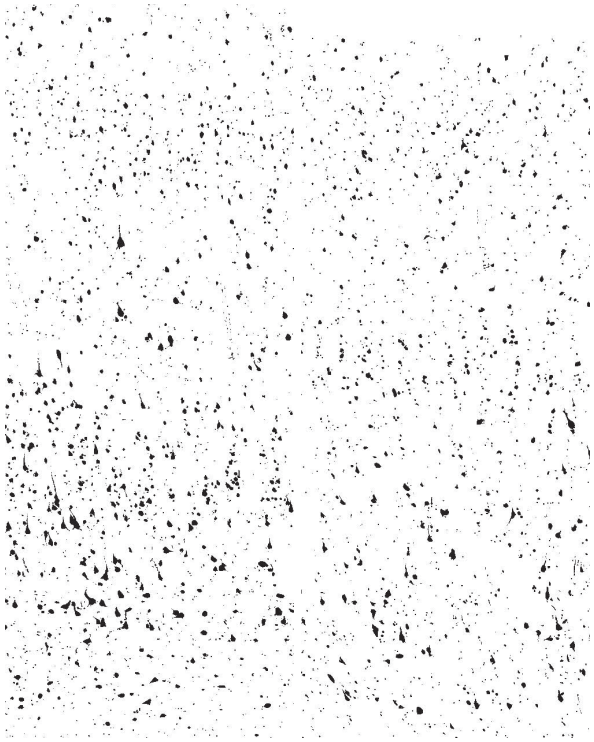
Figure 4.19: We compute histogram of orientated gradients in each layer. One little icon means there is a histogram in the layer. There are 42 histograms representing one image.

4.2.4 Stable Regions Feature Extraction

Maximally stable extremal regions are used for blob detection in images. In our sample images, the neurons are brown and bigger than the blue nuclei and we want to find out the disorders of neurons, so it makes sense for us to try to detect the neurons and then use their features for classification. We used the vl_mser function in the VLFeat open source library [56] to detect neurons from the whole image.

First we set the variables of vl_mser function properly so that we get as many regions as possible. The function returned the seeds of the stable regions. Each region can be identified uniquely by one of its pixel x, as the connected component of the level set which contains x. Such a pixel is called the seed of the region. We find the pixels inside the region by finding the connected component that contains the seed and get a binary image that specifies region/non-region membership. However, this may cause a problem that some region may be nested inside another region, as shown in Figure 4.22. To solve this problem we find all the seeds that appear in more than one region and then throw them away. Therefore the nested regions have been eliminated.

Afterwards we test several thresholds to see which one works the best for filtering out

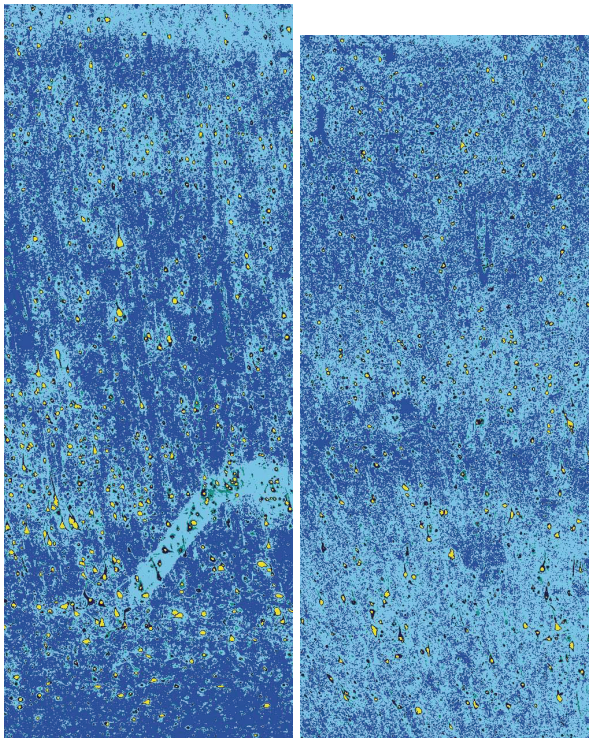


(a) (b)

Figure 4.20: The 2-bin intensity map of a normal sample and an abnormal sample. The black part contains most of the neurons. We can tell that the normal sample has denser neuron distribution and the layered structure is more obvious than the abnormal sample.

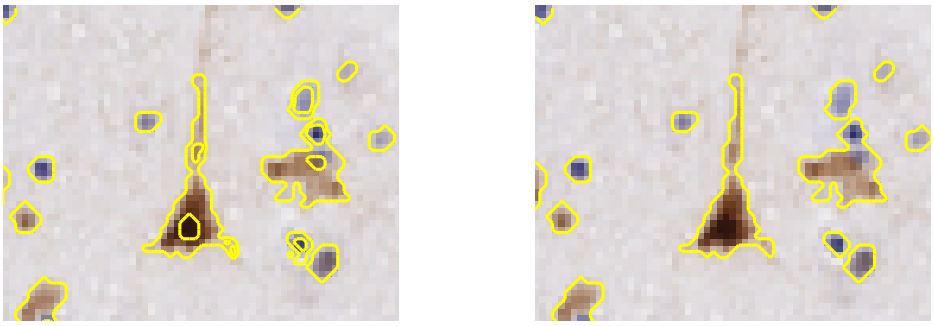
spurious regions, so that only regions corresponding to neurons are left. Since the neurons are brown and nuclei are blue, we computed the average of blue component in each region. We tried different thresholds of blue component and set it as 160. Any region with higher blue component than that is filtered. Also, we tried and set a threshold of region size as 40 since we expect neurons to have size larger than 40. We also set thresholds of region intensity. Figure 4.23 shows how the thresholding affects the stable region results.

After we have detected the stable regions, we used the properties of the stable regions as the image features. As usual, we detect stable region features in layers. For each layer, we extract 15 features which are: region number, average region size, average x and y width of boundingbox, average ConvexArea, average Extent, average FilledArea, average Perimeter, Max, min and mean intensity, histogram of orientation with 4 bins. The detailed descriptions of these features are shown in Table 4.1. These features give substantial information that reflect how the neurons are distributed in this layer. Finally we combined the features for each layer into a feature vector. As before, we use the laying in Figure 4.14. We have $15 \times 22 = 330$



(a) (b)

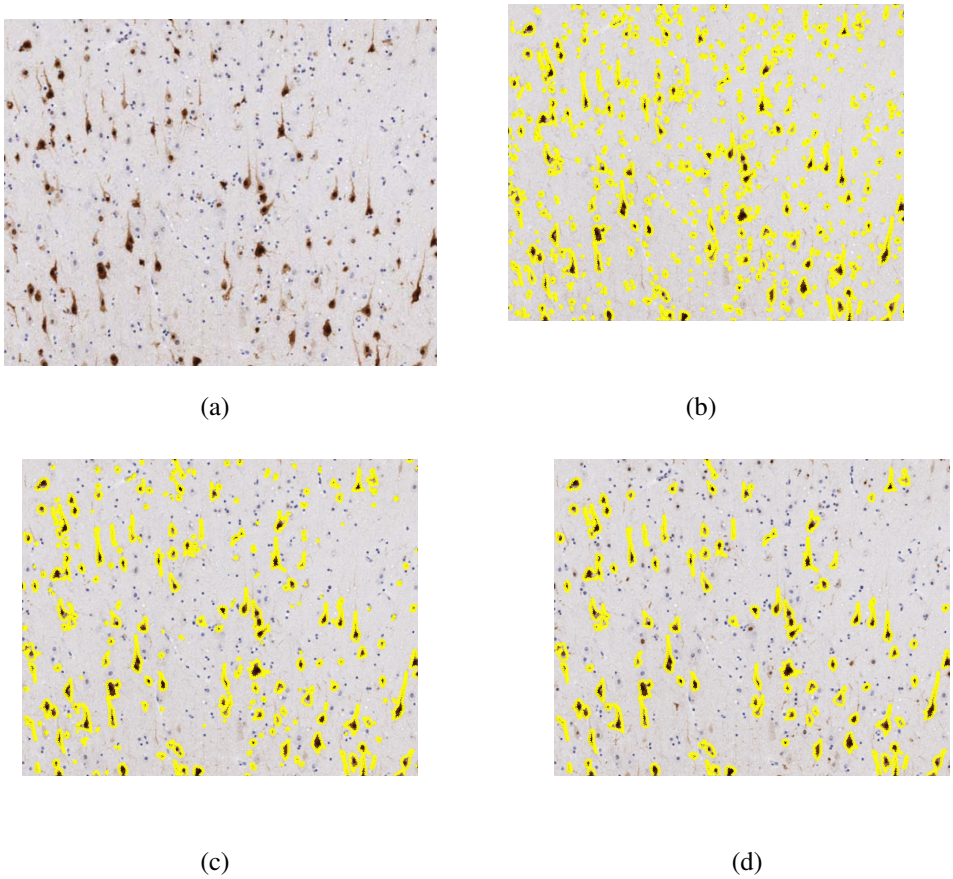
Figure 4.21: The 6-bin intensity map of a normal sample and an abnormal sample. Every bin is assigned a random color. The background is brightest in the original image so it corresponds to bin number 6. The background of normal sample is more noticeable while in the abnormal sample the background tends to be mixed with bin number 5.



(a) (b)

Figure 4.22: We fix the nested region problem.

stable region features for one image.



(a) (b)

(c) (d)

Figure 4.23: The stable region results with different thresholding. (b) is the the initial result. (c) is the result with threshold of blue component. (d) is the result with thresholds of blue component, size of region, intensity of region added. Only the big enough regions are kept. They mainly correspond to neurons.

4.3 Classification

After the feature extraction is done, we are ready to train classifiers on the feature vectors from the training set. The main method we used for classification is boosting. It is often used for binary classification task. We used the GML AdaBoost Toolbox v0.4 implemented by Vezhnevets [57]. The toolbox implemented 3 different boosting schemes: Real AdaBoost [44], Gentle AdaBoost [15] and Modest AdaBoost [58]. Real AdaBoost implemented the basic boosting algorithm introduced by Fruend and Schapire which is the "hardcore" boosting algorithm. Gentle AdaBoost is a more robust version of Real AdaBoost. It slightly outperforms Real Adaboost and considerably better on noisy data and more resistant to outliers. Modest AdaBoost is mostly aimed for resistance to overfitting. Modest AdaBoost performs better than

Feature number	Description
1	number of regions
2	average of Area size (of all regions in this layer, same below)
3	average of BoundingBox x width
4	average of BoundingBox y width
5	average size of ConvexHull
6	average of Extent
7	average size of FilledArea
8	average of Perimeter
9	average of max intensity
10	average of mean intensity
11	average of min intensity
12	histogram of orientation bin 1
13	histogram of orientation bin 2
14	histogram of orientation bin 3
15	histogram of orientation bin 4

Table 4.1: The descriptions of 15 stable regions features extracted in one layer. A BoundingBox is the smallest rectangle containing the region. A ConvexHull is the smallest convex polygon that can contain the region. Extent means the ratio of pixels in the region to pixels in the total BoundingBox. Size of FilledArea refers to the number of on pixels in FilledImage. A FilledImage means a binary image of the same size as the bounding box of the region with all holes filled in. Perimeter means the distance around the boundary of the region. Orientation of a region indicates the angle between the x-axis and the major axis of the ellipse that has the same second-moments as the region. We obtain a histogram of orientation with 4 bins from all the regions in the layer.

Real and Gentle AdaBoost in terms of test error and overfitting.

The weak learner used in the toolbox is a tree learner. A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. A decision tree is also named as classification tree or regression tree. In a decision tree structure, leaves represent class labels and branches represent conjunctions of features that lead to those class labels.

Figure 4.24 shows an example of a decision tree. It is a decision tree about whether we

should do some outdoor sports like tennis. Whether we should go depends on the conditions of weather, humidity and wind which correspond to the decision nodes of the tree. The decision tree in Figure 4.24 corresponds to the values shown in Table 4.2. In a decision tree, a node could have many values, like there could be many weather conditions. The branches correspond to the values of the tree node. The leaves correspond to the output, in this case, whether we should go or not. A decision can be converted into a binary decision tree, as shown in Figure 4.25. Each tree branch has a binary value.

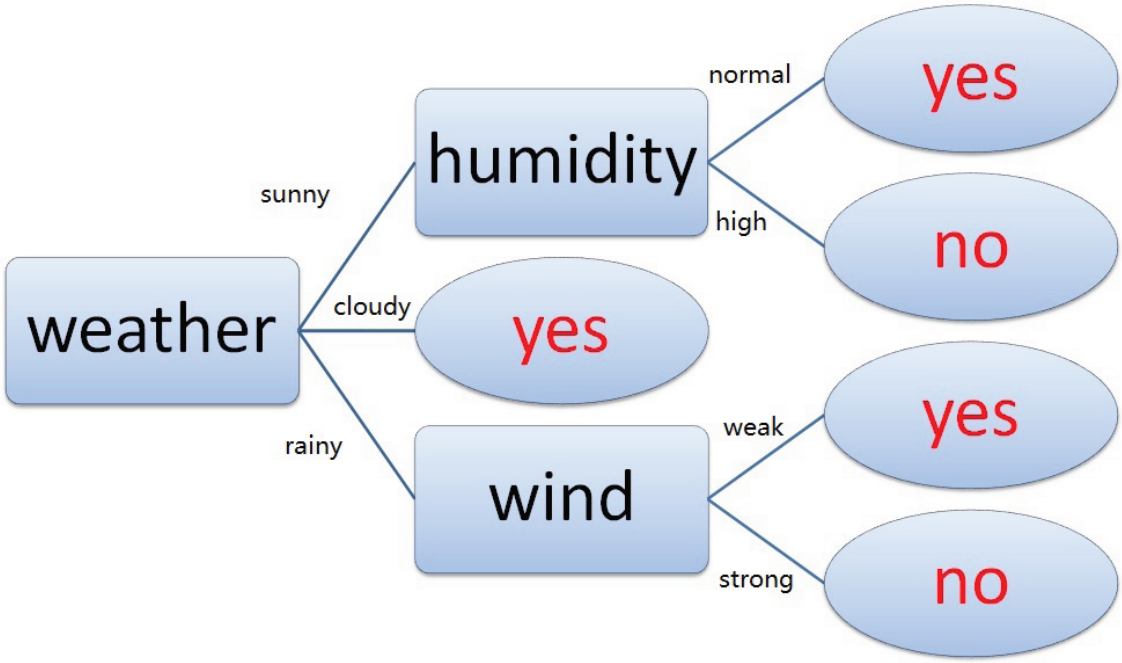


Figure 4.24: This is a decision tree about whether we should do outdoor sports. The tree nodes correspond to the conditions we consider. The branches correspond to the values of a tree node. The yes and no in the oval is the output of the decision tree. This tree corresponds the outputs shown in Table 4.2.

In the toolbox Classification and Regression Trees (CART) are built. A CART decision tree contains leaves representing the classification result and nodes representing the predicate. Branches of tree are marked true or false. Figure 4.26 shows an example of CART. Each node has a value of the threshold of the predicate. When we use the tree weak learner for classification, we start from the root and follow the nodes and branches until we reach a leaf. Assuming the dimension of our feature is n, each node of one tree learner corresponds a feature dimension. In our experiments, we set the number of tree splits to 3 so it corresponds to 3

Weather condition	Humidity	Wind	Whether we should go
sunny	normal	weak	yes
sunny	normal	strong	yes
sunny	high	weak	no
sunny	high	strong	no
rainy	normal	weak	yes
rainy	normal	strong	no
rainy	high	weak	yes
rainy	high	strong	no
cloudy	normal	weak	yes
cloudy	normal	strong	yes
cloudy	high	weak	yes
cloudy	high	strong	yes

Table 4.2: The outputs of whether we should go in every conditions about weather, humidity and wind.

feature dimensions and each tree has 4 leaves. Therefore each decision tree corresponds to 4 weak classifiers. There are two kinds of structure for decision trees with 3 splits, shown in Figure 4.27. In the toolbox, the boosting algorithm finds a threshold that separates normal and abnormal samples with least error for all n dimensions. Then it chose the dimension with the least error to construct a node of CART. It boosted the weak learners of CART to a strong learner.

We tested all the three boosting algorithm for classification. We performed leave-one-out cross-validation as well as 10-fold cross-validation on the dataset. We recorded the error rates of different iterations and observed how the classification results change with more iterations. We keep training the classifier until the cross-validation error start to increase. For different number of features, it takes different number of iterations for the error rate to converge. We compared the performance of the three boosting methods. Boosting is supposed to select the best features for classification, since the weak tree learners chosen have lowest error and each weak tree learner corresponds to feature dimensions. We save the constructed classifiers and we can analyze which features are selected by boosting. This can provide information to help us in the feature detection.

In our dataset there are many more normal samples than abnormal ones. The number of

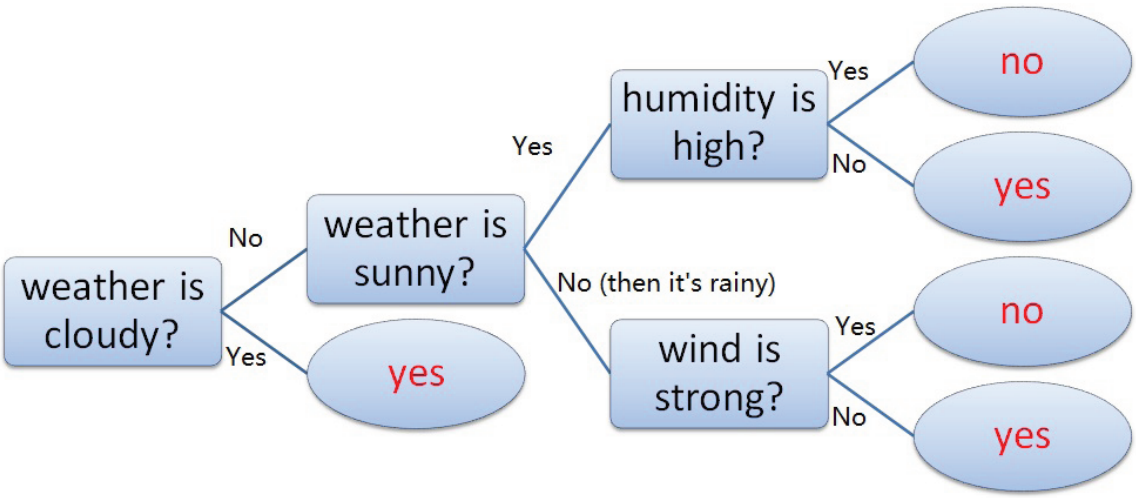


Figure 4.25: This is a binary decision tree with the same function of Figure 4.24. Each branch corresponds a binary value.

normal samples and abnormal samples in the training set should be roughly the same, in order to form a balanced training set. In a binary classification problem, with imbalanced training data, classification rules that predict the majority class tend to be stronger than those that predict the minority class. Therefore, test samples in the minority class are misclassified more often than those in the majority class. Standard classifiers usually perform poorly on imbalanced datasets. To avoid imbalanced training set, we randomly chose as many normal as there are abnormal samples to form the training set. We trained the classifier, performed the cross-validation on the training set and also used the trained classifier to test on the other samples. We performed the classification on all the three datasets: CorticalProfiles_NeuN, corticalQuantHist and H&E&NEUN stain. We marked the ground truth and classification result of samples on the whole profile like figure to visualize where the error occurs. This will help the pathologists for the future research.

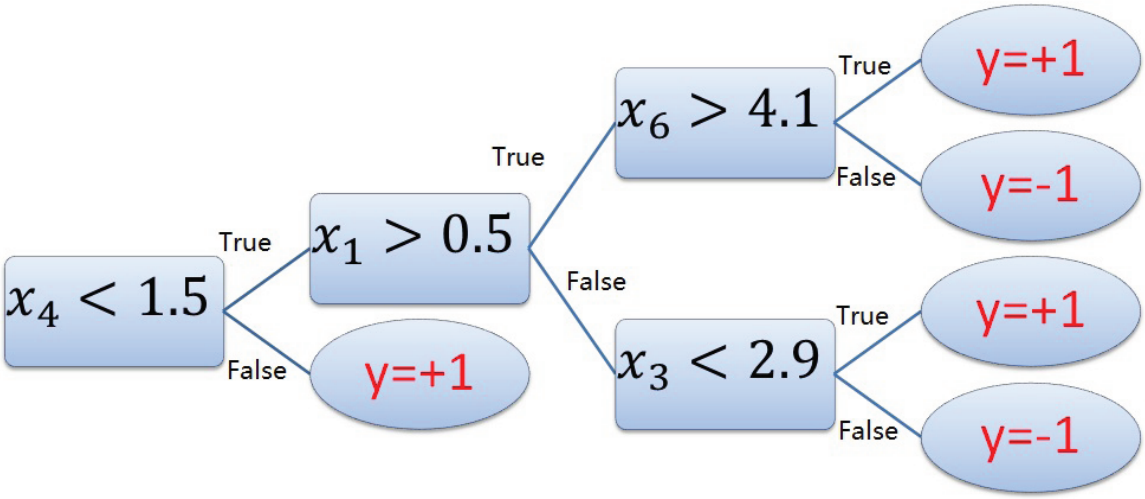
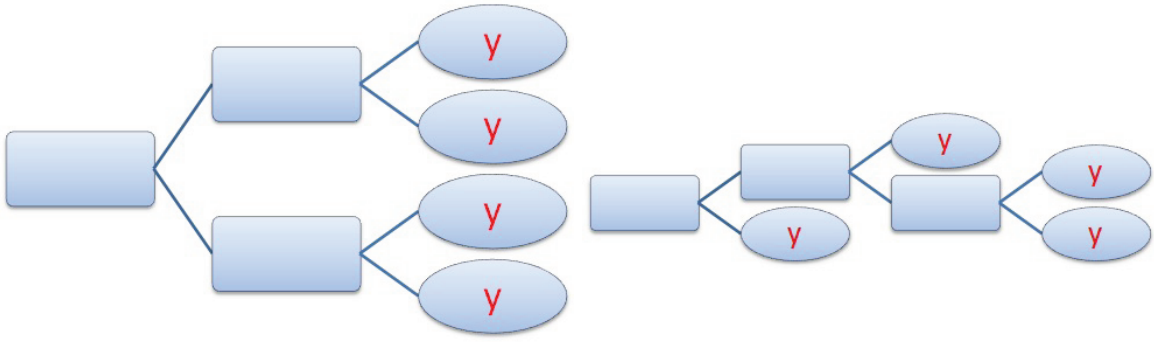


Figure 4.26: This is a CART example. The branch has a boolean value. Each node corresponds a threshold of one feature.



(a) (b)

Figure 4.27: Two structures for decision tress with 3 splits.

Chapter 5

Experimental Results

In this chapter we present the classification results on all the three datasets. We tried different kinds of image features and different classification methods on the datasets. We will compare their different performance.

5.1 Classification on CorticalProfiles_NeuN Dataset

In this dataset, there are samples from two patients, EPI_P006 and EPI_P014. All the sample images have the width of 500 but different heights. The samples in EPI_P014 are labeled as normal or abnormal. There are 67 abnormal samples and 257 normal samples. All the samples in EPI_P006 are considered as normal. From EPI_P014, We randomly selected 67 normal samples and combined them with the abnormal samples to form the training set. We performed 10-fold cross-validation on the dataset of 134 samples. The samples are shuffled randomly are divided equally into 10 folds. The features we extracted include SIFT, HOG and histogram of color. We extracted SIFT features of bin size 3, 5 and 8 from every 2 pixels. Then for each bin size, the features from training set are clustered in to k clusters. We set k as 50, 100, 200 and 500. We obtained histogram of representative features for each layer. The number of layer is set to 1, 3, 6 and 12. Therefore, the total number of SIFT features for one image is 56100. For color features, we clustered RGB colors into 3, 5 and 7 clusters. For each layer we obtain 15 features and for one image there are 330 color features. For HOG features we set the orientation channels as 4, 8 and 16 and extracted histograms from layers. The layer number is set to 2, 4, 6, 8, 10 and 12. We combined all the features in to a feature vector. Figure 5.1 shows all the features we used in this experiment. The feature vector we used for boosting has a dimension of 57606. As shown in Table 5.1, the accuracy of 10-fold cross-validation is

91.0%. We can see from Figure 5.2 that with number of iterations increased, the error rate goes down and the error rate converges after 400 iterations.

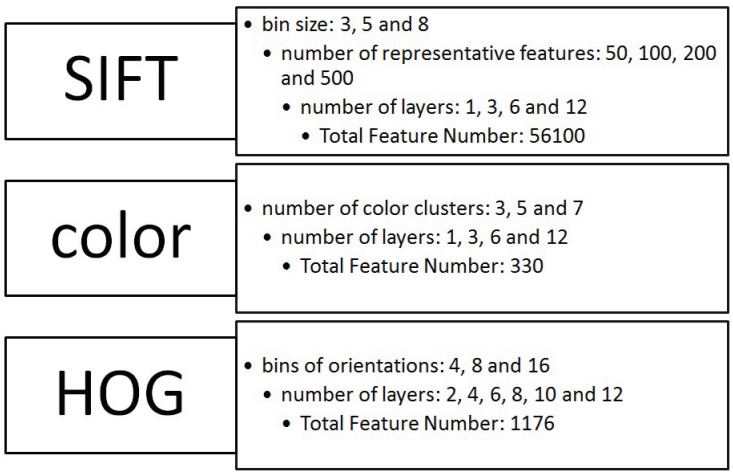


Figure 5.1: All the features detected in the experiment on CorticalProfiles_NeuN dataset. The total number of features is 57606.

Evaluation Method	10-fold cross-validation
Dataset	CorticalProfiles_NeuN
Training Data	EPI_P014
Number of Samples	134
Image Feature	SIFT, HOG, histogram of color
Number of Features	57606
Classification Method	Gentle AdaBoost
Number of Iterations	2000
Accuracy Rate	91.0%

Table 5.1: The 10-fold cross-validation result on the CorticalProfiles_NeuN dataset.

When the feature dimension is high, the classification using boosting is computationally expensive. For the experiment in Table 5.1, it takes 30 minutes for training one time and the 10-fold cross-validation cost 5 hours. The classifier constructed by boosting can be saved in a file and we tried to analyze which features did boosting selected. We found that SIFT bin 3 and bin 5 features are selected more often than bin 8 features. We tried to use subvectors of the feature vector for nearest neighbor classification. Therefore, we used the histograms of

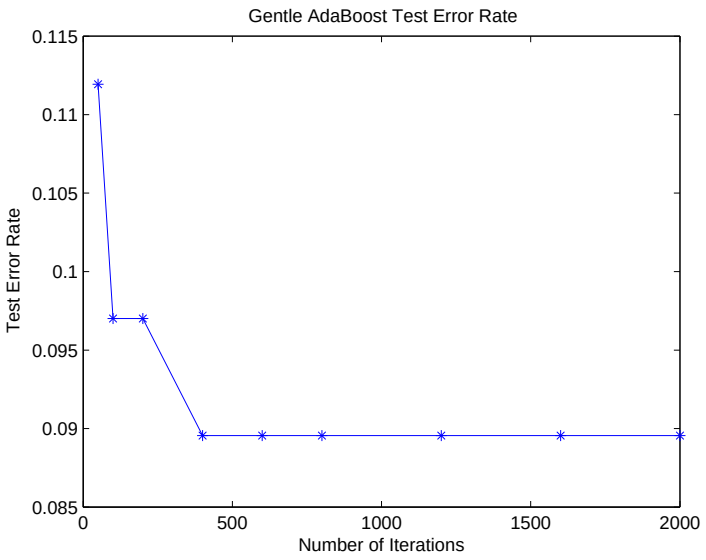


Figure 5.2: The curve of error rate of the 10-fold cross-validation experiment in Table 5.1. Error rate goes down when number of iterations increased.

representative features of bin 3 SIFT with different number of layers and clusters. The error rate is shown in Table 5.2. When we use the histograms with 200 bins and partition the image into 6 layers, we have the best accuracy rate of 93.3%. In this case, there are 1200 features for one image. However, the drawback of nearest neighbor method is that it has high computational complexity.

Number of clusters\Number of layers	1	3	6	12
50	86.6%	88.8%	91.0%	91.8%
100	91.0%	89.6%	92.5%	89.6%
200	91.0%	91.0%	93.3%	90.3%
500	91.8%	91.0%	92.5%	91.8%

Table 5.2: The nearest neighbor accuracy rate using SIFT features bin 3 with different number of layers and clusters.

5.2 Classification on corticalQuantHist Dataset

In the corticalQuantHist dataset there are samples of three patients with epilepsy, EPI_P006, EPI_P014 and EPI_P015. The samples in EPI_P014 have normal or abnormal labels. This dataset is different from CorticalProfiles_NeuN dataset in some ways. The profile images are sampled from different position of the full profile. For EPI_P014, 5 more full profiles are added. Also, the position of each sliced profile on the whole profile are given. Most of the samples in this dataset has profile layers labeled, those samples appear on the neuron density maps. We can locate the misclassified samples on the neuron density maps and find the relation between misclassification and neuron densities.

5.2.1 Histogram of Color

There are 67 abnormal samples in EPI_P014. We randomly selected 67 normal samples and form a training set with 134 samples. Instead of SIFT, we tried histogram of color first. We performed 10-fold cross-validation. We extracted 15 features per layer and 330 features per image. We used Gentle AdaBoost for classification. As shown in Table 5.3 and Figure 5.3, the accuracy rate is 90.3%. When we add histograms of 24 layers, the accuracy rate is 87.3%, as shown in Table 5.4 and Figure 5.4. To obtain histogram of color features are computationally efficient. Since the number of features is not large, it is also computationally efficient for training classifiers with boosting. The 10-fold cross-validation computataion time is 50 seconds.

5.2.2 Histogram of Normalized Intensity

Our next trial of feature is histogram of normalized intensity. As we explained before, we divided the intensity space into uniform channels after normalizing the image. We set the number of channels as 2, 3, 4, 5 and 6. The result are shown in Table 5.5 and Figure 5.5. We got 91.8% of accuracy. Error rate converges when number of iterations increased to 200. We performed leave-one-out cross-validation using histogram of normalized intensity features, as shown in Table 5.6, the accuracy is 92.5%. It gives better result because leave-one-out cross-validation is a more accurate estimation of classification than 10-fold cross-validation. We also tried histogram of intensity on images without normalization, the accuracy of 10-fold cross-validation is 89.6%, not as good as the former case.

Computing histograms of normalized intensity is computationally efficient. The boosting is efficient as well. The computation time is 60 seconds.

Evaluation Method	10-fold cross-validation
Dataset	corticalQuantHist
Training Data	EPI_P014
Number of Samples	134
Image Feature	Histogram of color
Number of Bins	3, 5, 7
Number of Layers	1, 3, 6, 12
Number of Features	330
Classification Method	Gentle AdaBoost
Number of Iterations	1000
Accuracy Rate	90.3%

Table 5.3: The 10-fold cross-validation result on the corticalQuantHist dataset using histogram of color features.

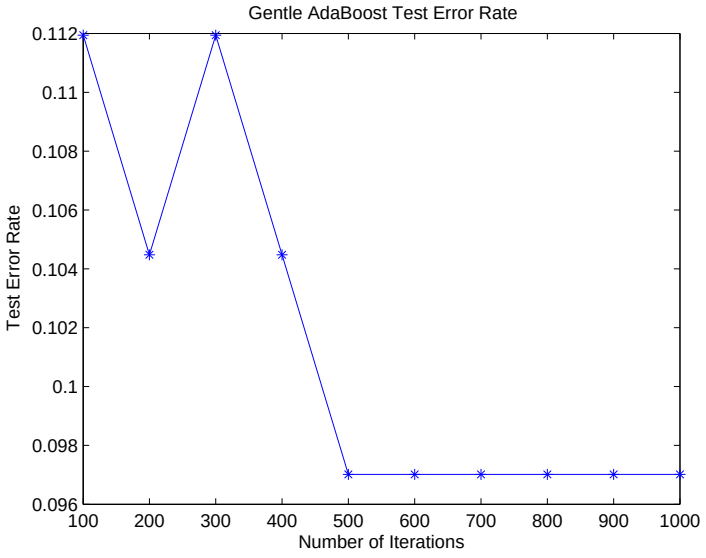


Figure 5.3: The curve of error rate of 10-fold cross-validation experiment in Table 5.3. Error rate converges when number of iterations increased to 1000.

We obtained the training error of experiments using histogram of normalized intensity features. For all the three boosting methods, the training errors went to 0 after 20 iterations.

In order to visualize the result to see where the misclassified samples appear on the neuron

Evaluation Method	10-fold cross-validation
Dataset	corticalQuantHist
Training Data	EPI_P014
Number of Samples	134
Image Feature	Histogram of color
Number of Bins	3, 5, 7
Number of Layers	1, 3, 6, 12, 24
Number of Features	330
Classification Method	Gentle AdaBoost
Number of Iterations	1000
Accuracy Rate	87.3%

Table 5.4: The 10-fold cross-validation result on the corticalQuantHist dataset using histogram of color features. Features of 24 layers are added.

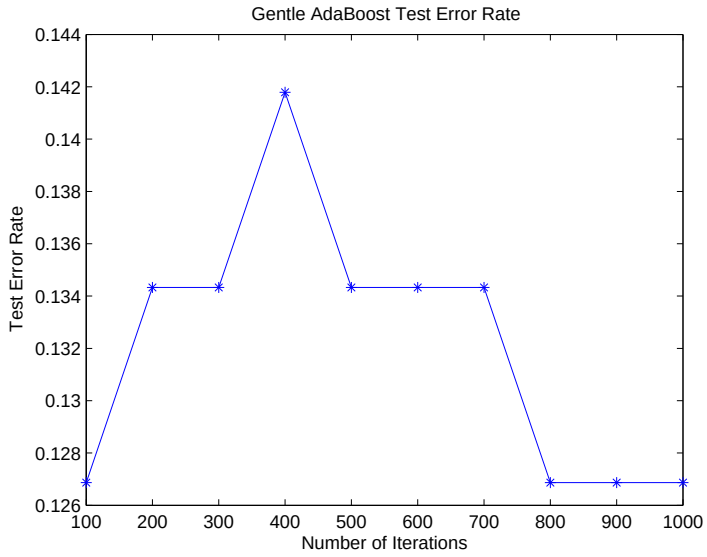


Figure 5.4: The curve of error rate of 10-fold cross-validation experiment in Table 5.4. Error rate converges when number of iterations increased to 1000.

density map, we used the samples with layers labeled. There are 59 abnormal samples with layers so we form a training set with 118 samples with an identical number of normal samples. We still used histogram of normalized intensity. As shown in Table 5.7, the accuracy is 90.7%.

Evaluation Method	10-fold cross-validation
Dataset	corticalQuantHist
Training Data	EPI_P014
Number of Samples	134
Image Feature	Histogram of normalized intensity
Number of Bins	2, 3, 4, 5, 6
Number of Layers	1, 3, 6, 12
Number of Features	440
Classification Method	Gentle AdaBoost
Number of Iterations	200
Accuracy Rate	91.8%

Table 5.5: The 10-fold cross-validation result on the corticalQuantHist dataset using histogram of normalized intensity features.

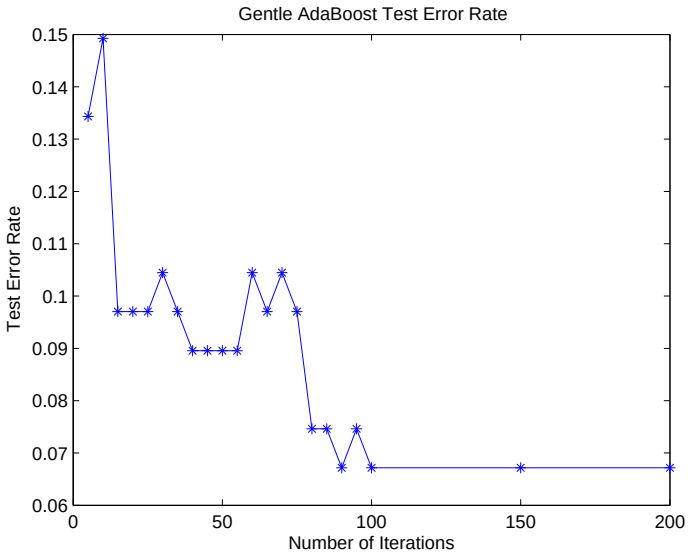


Figure 5.5: The curve of error rate of 10-fold cross-validation experiment in Table 5.5. Error rate converges when number of iterations increased to 200.

It is slightly worse than the experiment in Table 5.5, since there are less abnormal samples. In the neuron density maps in Figure 5.6, the ground truth is marked by adding a circle around the sample. The red circle means abnormal sample and black circle means normal sample. The

Evaluation Method	leave-one-out cross-validation
Dataset	corticalQuantHist
Training Data	EPI_P014
Number of Samples	134
Image Feature	Histogram of normalized intensity
Number of Bins	2, 3, 4, 5, 6
Number of Layers	1, 3, 6, 12
Number of Features	440
Classification Method	Gentle AdaBoost
Number of Iterations	200
Accuracy Rate	92.5%

Table 5.6: The leave-one-out cross-validation result on the corticalQuantHist dataset using histogram of normalized intensity features.

misclassified sample have a big asterisk on it. We can tell that the misclassified samples often appear on the boundaries of contours. We used the classifier trained by boosting in Table 5.7 to test on the other two patients EPI_P006 and EPI_P015, the 55 samples are all classified as normal, which they should be since the pathologists did not see any abnormalities in them.

We also performed SVM using histogram of normalized intensity on the samples with layers labeled in EPI_P014. The 10-fold cross-validation accuracy of linear SVM is 91.5%, shown in Table 5.8.

5.2.3 Stable Regions

Stable region features are also used in our classification. As we described in Chapter 4, we detected the maximally stable extremal regions in the whole image and extracted features from the stable regions. We extracted 15 features per layer and 330 features per image. The result is shown in Table 5.9. The accuracy of the classification is 93.2%. This is the highest accuracy we have got on this dataset. We saved the classifier trained by boosting in a file and see which features are selected by boosting. The result shows that the number of regions in a layer, the region size and the intensities inside the region are more important. They give more information useful for classification. The toolbox [56] we used for stable regions detection is a very efficient algorithm.

Evaluation Method	10-fold cross-validation
Dataset	corticalQuantHist
Training Data	The samples with layers labeled in EPI_P014
Number of Samples	118
Image Feature	Histogram of normalized intensity
Number of Bins	2, 3, 4, 5, 6
Number of Layers	1, 3, 6, 12
Number of Features	440
Classification Method	Gentle AdaBoost
Number of Iterations	200
Accuracy Rate	90.7%

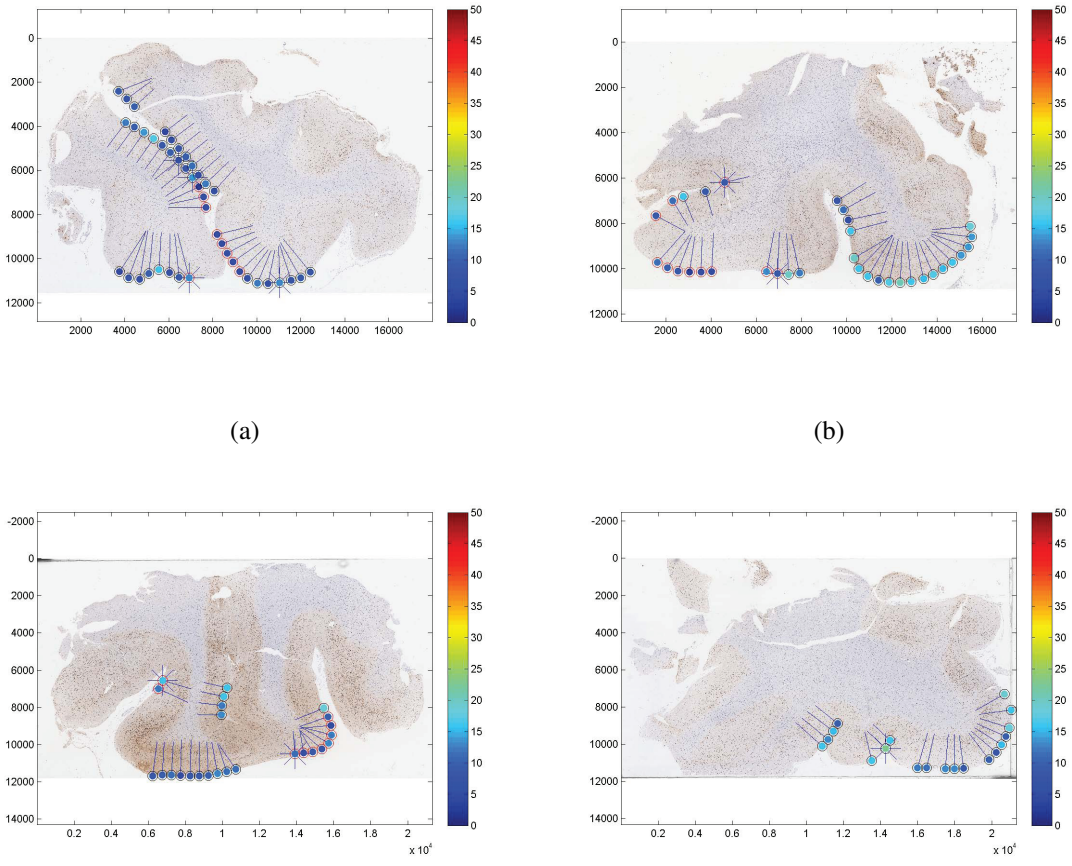
Table 5.7: The 10-fold cross-validation result on the corticalQuantHist dataset using histogram of normalized intensity features. We used the samples with layer labeled as our dataset.

Evaluation Method	10-fold cross-validation
Dataset	corticalQuantHist
Training Data	EPI_P014
Number of Samples	134
Image Feature	Histogram of normalized intensity
Number of Bins	2, 3, 4, 5, 6
Number of Layers	1, 3, 6, 12
Number of Features	440
Classification Method	Linear SVM
Accuracy Rate	91.5%

Table 5.8: The 10-fold cross-validation result on the corticalQuantHist dataset using histogram of normalized intensity features. We used SVM as the classifier.

5.3 Classification on H&E&NEUN stain Dataset

The stained data of EPI_P014 has the same ground truth with its corresponding in corticalQuantHist dataset. We cannot detect neurons from this dataset. We used histogram of



(a) (b)

(c) (d)

Figure 5.6: Three neuron density maps. Brighter dot means denser neuron. The misclassified samples are marked an asterisk.

normalized intensity features. As shown in Table 5.10, the best accuracy we can get is 80.5%. The result for stained data is not as good as the other datasets.

Evaluation Method	10-fold cross-validation
Dataset	corticalQuantHist
Training Data	The samples with layers labeled in EPI_P014
Number of Samples	118
Image Feature	Stable regions features
Number of Layers	1, 3, 6, 12
Number of Features	330
Classification Method	Gentle AdaBoost
Number of Iterations	200
Accuracy Rate	93.2%

Table 5.9: The 10-fold cross-validation result on corticalQuantHist dataset using stable regions features.

Evaluation Method	10-fold cross-validation
Dataset	H&E&NEUN stain
Training Data	EPI_P014
Number of Samples	118
Image Feature	Histogram of normalized intensity
Number of Bins	2, 3, 4, 5, 6
Number of Layers	1, 3, 6, 12
Number of Features	440
Classification Method	Real AdaBoost
Number of Iterations	100
Accuracy Rate	80.5%

Table 5.10: The experiment result on the H&E&NEUN stain dataset using histogram of normalized intensity features.

Chapter 6

Conclusion

In this thesis, we apply computer vision and machine learning techniques to tackle the problem of epilepsy lesion classification. Our goal is to validate the use of MRI for localizing seizure focus for improved surgical planning. We learn to detect histology brain image slices that are associated with epilepsy lesions. Focal cortical dysplasia (FCD) is the most common pathology in MRI-negative epilepsy and classification of FCD is needed for improved MRI validation. We are interested in one kind of focal cortical dysplasia, which is Type 1 FCD.

Datasets of histology brain samples are obtained by medical imaging scientists and provided to us. Some of the samples show abnormalities that are considered as Type 1 FCD by pathologists. We evaluate a variety of image feature types that are popular in computer vision community to find those features that are appropriate for the epilepsy lesion classification. We test Boosting, SVM and the Nearest Neighbor methods to train and classify the images into normal and lesion ones. We also performed automatically neuron detection.

In the experiments we performed the classification on three different datasets. Since the FCD is related to disorder in neuron layers, we partition images into uniform layers and combined the features in each layer to construct the feature vector. For most of the experiments, we obtain at least 90.0% of accuracy. The best accuracy we get is 93.3% when using SIFT features and nearest neighbor for classification. We applied MSER detection to detect neurons and obtain a good result quantifying the neuron density. The accuracy of the experiment using boosting for classification and stable region features is 93.2%, which is the second best result we obtained for all the experiments. These are satisfying results for this experiment. When using feature vectors with lower dimension and AdaBoost for training, the experiment is computational efficient. It takes less than 60 seconds for the 10-fold cross-validation to execute.

In our work we present effective and efficient methods for classifying Type 1 FCD and also

quantifying neuron densities. The results of the classification and quantification are promising. The methods are computational efficient, both for the feature extraction and the classification. In most cases, it just takes 6 seconds for one training experiment and 60 seconds for the 10-fold cross-validation. The trained classifier can be applied to more samples in the future. It takes 5 seconds to detect all the regions that are likely to be neurons for one histology sample image. As far as we know, our work of performing histology image classification and automatic quantification of focal cortical dysplasia in the correlation study of MRI and epilepsy histopathology is the first of its kind. Our method could potentially provide useful information for surgical planning.

More work could be done in the future. We could try more image features like those based on texture and we can try different classifiers as well. Also, automatic layering of the profile images could be really useful for surgical planing. In our work, we divided the histology sample image into uniform horizontal layers and we don't use the labeled layering information. In fact, the layering does not have to be horizontal but corresponding to actual boundaries between layers. The boundaries between layers may not be lines, but curves. Also, we could set a better filter for the neuron detection if we know more about the properties of neurons. Since it is really difficult to obtain the images we use in the experiment, there is only one patient whose profiles show abnormalities and we performed our classification on only one patient. The accuracy rate would be more convincing if we have samples from more patients.

Bibliography

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *Knowledge and Data Engineering, IEEE Transactions on*, 17(6):734–749, 2005.

[2] E. Aleskerov, B. Freisleben, and B. Rao. Cardwatch: A neural network based database mining system for credit card fraud detection. In *Computational Intelligence for Financial Engineering (CIFEr), 1997., Proceedings of the IEEE/IAFE 1997*, pages 220–226. IEEE, 1997.

[3] I. Bankman. *Handbook of medical imaging: processing and analysis management*. Academic Press, 2000.

[4] A. Blake and M. Isard. 3D position, attitude and shape input using video tracking of hands and lips. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 185–192. ACM, 1994.

[5] I. Blümcke, M. Thom, E. Aronica, D.D. Armstrong, H.V. Vinters, A. Palmini, T.S. Jacques, G. Avanzini, A.J. Barkovich, G. Battaglia, et al. The clinicopathologic spectrum of focal cortical dysplasias: A consensus classification proposed by an ad hoc task force of the ilae diagnostic methods commission1. *Epilepsia*, 52(1):158–174, 2011.

[6] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[7] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[8] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[9] N. Dalal, B. Triggs, and C. Schmid. Human detection using oriented histograms of flow and appearance. *Computer Vision–ECCV 2006*, pages 428–441, 2006.

[10] M. Donoser and H. Bischof. Efficient maximally stable extremal region (mser) tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 553–560. IEEE, 2006.

[11] SH Eriksson, SL Free, M. Thom, L. Martinian, MR Symms, TM Salmenpera, AW McEvoy, W. Harkness, JS Duncan, and SM Sisodiya. Correlation of quantitative mri and neuropathology in epilepsy surgical resection specimensłt2 correlates with neuronal tissue in gray matter. *Neuroimage*, 37(1):48, 2007.

[12] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 524–531. IEEE, 2005.

[13] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010.

[14] Y. Freund and R. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory*, pages 23–37. Springer, 1995.

[15] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). *The annals of statistics*, 28(2):337–407, 2000.

[16] L.J. Garey. Brodmanns localisation in the cerebral cortex. *London: Smith-Gordon*, 135, 1994.

[17] P. Gehler and S. Nowozin. On feature combination for multiclass object classification. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 221–228. IEEE, 2009.

[18] A.M.G. Gillies. Machine learning procedures for generating image domain feature detectors. *Dissertation Abstracts International Part B: Science and Engineering[DISS. ABST. INT. PT. B- SCI. & ENG.],*, 46(7), 1986.

[19] J. Goodman, G.V. Cormack, and D. Heckerman. Spam and the ongoing battle for the inbox. *Communications of the ACM*, 50(2):24–33, 2007.

[20] I. Gordon and D. Lowe. What and where: 3D object recognition with accurate pose. *Toward category-level object recognition*, pages 67–82, 2006.

[21] M. Goubran, A.R. Khan, C. Crukley, S. Buchanan, B. Santyr, T.M. Peters, et al. Robust registration of sparsely sectioned histology to ex-vivo mri of temporal lobe resections. In *SPIE Medical Imaging*, pages 83141V–83141V. International Society for Optics and Photonics, 2012.

[22] Z.S. Harris. Distributional structure. *Word*, 1954.

[23] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3D modeling of indoor environments. In *the 12th International Symposium on Experimental Robotics (ISER)*, volume 20, pages 22–25, 2010.

[24] D. Huber, A. Kapuria, R. Donamukkala, and M. Hebert. Parts-based 3D object classification. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–82. IEEE, 2004.

[25] O. Javed and M. Shah. Tracking and object classification for automated surveillance. *Computer VisionłECCV 2002*, pages 439–443, 2006.

[26] L.P. Kaelbling, M.L. Littman, and A.W. Moore. Reinforcement learning: A survey. *arXiv preprint cs/9605103*, 1996.

[27] Y. Ke, R. Sukthankar, and M. Hebert. Efficient visual event detection using volumetric features. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 166–173. IEEE, 2005.

[28] V. Kolmogorov and R. Zabih. Multi-camera scene reconstruction via graph cuts. *Computer VisionłECCV 2002*, pages 8–40, 2002.

[29] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

[30] Y. LeCun, LD Jackel, L. Bottou, C. Cortes, J.S. Denker, H. Drucker, I. Guyon, UA Muller, E. Sackinger, P. Simard, et al. Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural networks: the statistical mechanics perspective*, 261:276, 1995.

[31] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900. IEEE, 2002.

[32] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[33] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[34] J. MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

[35] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[36] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L.V. Gool. A comparison of affine region detectors. *International journal of computer vision*, 65(1):43–72, 2005.

[37] R.T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of*, pages 144–155, 1994.

[38] N.M. Oliver, B. Rosario, and A.P. Pentland. A bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):831–843, 2000.

[39] A. Palmini, I. Najm, G. Avanzini, T. Babb, R. Guerrini, N. Foldvary-Schaefer, G. Jackson, HO Lüders, R. Prayson, R. Spreafico, et al. Terminology and classification of the cortical dysplasias. *Neurology*, 62(6 suppl 3):S2–S8, 2004.

[40] D.L. Pham, C. Xu, and J.L. Prince. Current methods in medical image segmentation 1. *Annual review of biomedical engineering*, 2(1):315–337, 2000.

[41] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. *Computer Vision–ECCV 2006*, pages 430–443, 2006.

[42] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3D object modeling and recognition using affine-invariant patches and multi-view spatial constraints. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–272. IEEE, 2003.

[43] A.L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 44(1.2):206–226, 2000.

[44] R.E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine learning*, 37(3):297–336, 1999.

[45] S. Se, D. Lowe, and J. Little. Vision-based mobile robot localization and mapping using scale-invariant features. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 2051–2058. IEEE, 2001.

[46] J. Shi and J. Malik. Normalized cuts and image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):888–905, 2000.

[47] J. Sivic, B.C. Russell, A.A. Efros, A. Zisserman, and W.T. Freeman. Discovering objects and their location in images. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 370–377. IEEE, 2005.

[48] J. Sivic and A. Zisserman. Efficient visual search of videos cast as text retrieval. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):591–606, 2009.

[49] C. Stauffer and W.E.L. Grimson. Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757, 2000.

[50] R.S. Sutton. Temporal credit assignment in reinforcement learning. 1984.

[51] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 1. Cambridge Univ Press, 1998.

[52] DC Taylor, MA Falconer, CJ Bruton, and JAN Corsellis. Focal dysplasia of the cerebral cortex in epilepsy. *Journal of Neurology, Neurosurgery & Psychiatry*, 34(4):369–387, 1971.

[53] G. Tur, R.E. Schapire, and D. Hakkani-Tur. Active learning for spoken language understanding. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, volume 1, pages I–276. IEEE, 2003.

[54] M.A. Turk and A.P. Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991.

[55] P.A. Van den Elsen, E.J.D. Pol, and M.A. Viergever. Medical image matching-a review with classification. *Engineering in Medicine and Biology Magazine, IEEE*, 12(1):26–39, 1993.

[56] A. Vedaldi and B. Fulkerson. VLFeat: An open and portable library of computer vision algorithms. `http://www.vlfeat.org/`, 2008.

[57] A. Vezhnevets. Gml adaboost matlab toolbox. *Technique Manual, Graphics and Media Laboratory, Computer Science Department, Moscow State University, Moscow, Russian Federation*, 2006.

[58] A. Vezhnevets and V. Vezhnevets. Modest adaboost-teaching adaboost to generalize better. *Graphicon-2005. Novosibirsk Akademgorodok, Russia*, 2005.

[59] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001.

[60] P. Viola, M.J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63(2):153–161, 2005.

[61] J. Von Oertzen, H. Urbach, S. Jungbluth, M. Kurthen, M. Reuber, G. Fernandez, and CE Elger. Standard magnetic resonance imaging is inadequate for patients with refractory focal epilepsy. *Journal of Neurology, Neurosurgery & Psychiatry*, 73(6):643–647, 2002.

[62] C. Wang and L. Guan. Graph cut video object segmentation using histogram of oriented gradients. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 2590–2593. IEEE, 2008.

[63] C.C. Wang and K.C. Wang. Hand posture recognition using adaboost with sift for human robot interaction. *Recent progress in robotics: viable robotic service to human*, pages 317–329, 2008.

[64] J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(2):210–227, 2009.

Curriculum Vitae

Name:	Junwei Sun
Post-Secondary Education and Degrees:	Peking University Beijing, China 2007 - 2011 B.Sc. University of Western Ontario London, ON 2011 - 2012 M.Sc. Supervisor: Dr. Olga Veksler
Related Work Experience:	Teaching Assistant The University of Western Ontario 2011 - 2012 Research Assistant The University of Western Ontario 2011 - 2012